SOCIAL NETWORKING: SECURITY, PRIVACY, AND APPLICATIONS

BY

SONIA JAHID

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2013

Urbana, Illinois

Doctoral Committee:

Associate Professor Nikita Borisov, Chair
Professor Carl A. Gunter, Co-chair
Associate Professor Indranil Gupta
Assistant Professor Apu Kapadia, Indiana University Bloomington

ABSTRACT

Online social networks have become ubiquitous and changed the way that users interact online. There has been an enormous growth in the usage of online social networking in the past few years as users share a variety of information including personal profiles, pictures, and messages to socialize with their friends in the Internet. Besides, several special purpose social networks have emerged to serve their users with useful functionalities. This vast amount of user data is valuable, and therefore, introduce several security and privacy risks and challenges.

In this thesis we propose several techniques to enhance the security and privacy features of online social networks. Our goal is to shift the control over user data from a centralized social network provider to the end users. We realize this concept by decentralization of the social networking architecture. First, we construct and implement a cryptographic access control mechanism that ensures data confidentiality and integrity, and efficiently supports the fine-grained access policies expected by social network users. Next, we present a detailed design of a decentralized online social network that focuses on security and privacy. Finally, we propose and implement auditable anonymity, a cryptographic scheme, which allows a social networking user to keep track of who accesses her data even when her data is encrypted and decentralized, without revealing this information to the storage provider.

To my family

ACKNOWLEDGMENTS

This dissertation would not have been possible without the help of my advisor Nikita Borisov. Nikita supported me when I needed it the most. I would like to thank Nikita for guiding me through this challenging journey.

Thanks to my fellow students and colleagues at University of Illinois at Urbana-Champaign who helped me with discussions, constructive suggestions, and advice every now and then. Also, thanks to my co-authors for the hard work and involvement in the projects that we did together.

Finally, thanks to my family and friends for their love and support. Thanks Imranul Hoque – my husband, peer, and friend for the love and support you have given me. Thanks to my son Shopnil Aydin whose birth changed my perspective about life and taught me how to accomplish what I want even though priorities of life change. Finally, thanks to my parents Rahima Nabi and Md. Eunus Jahid for everything that you have done and have been doing for me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

Over the past few years, Online Social Networks (OSNs) such as Facebook, Google+, Twitter, etc. have become ubiquitous and have changed the way users interact on the Internet. They provide a platform where users connect and share personal information with their social contacts, also known as *friends*. In addition to socialization, OSNs such as PatientsLikeMe, LinkedIn, Academia.edu serve their users with useful functionalities, for example, suggestions on a medical condition, jobs, and connection with research communities. A significant growth has occurred in the usage of OSNs in recent years. Started in 2004, Facebook in 2013 has over a billion active users [1]; Google+ has 359 million active users [2]; there are more than 500 million users on Twitter [3], and more than 225 million users on LinkedIn [4].

Along with services, either socialization or other functionalities, OSNs introduce new challenges and risks. Study shows that Facebook users share more than 30 billion pieces of data each month, whereas Twitter users send over 400 million tweets each day [1, 5]. This large amount of data is valuable [6], and introduces security and privacy risks, and turns the provider into an attractive attack target. While information such as a status post on Facebook may not be considered sensitive, social networks contain other information that are considered sensitive, and if revealed to inappropriate parties, can cause harm to the users. For example, recently, users have started using the location services to share their geographic information on the OSN. An ambiguous privacy setting may reveal this information to un-

wanted parties. Besides, a collection or this kind of information can be used to infer user's location [7, 8].

If not protected properly, social networking data can be used to infer sensitive information. For example, study has shown that social networking data has been used to derive social security number [9, 10]. OSNs may not store social security numbers directly, but they store enough data that can be used to derive such information. Users store their date of birth and email address that associate the users to their schools or specific communities. Hence, if revealed to inappropriate parties, this information may be used against the user to perform identity theft and other harmful activities.

Experimental and real phishing attacks have been performed on Facebook [11, 12]. Insiders can also intentionally or accidentally release private information [10, 13]. Several privacy compromises by the OSN provider either intentionally or accidentally also have thrown these issues into sharp focus [14, 15]. In addition to the OSN provider and the users in the OSN, third party applications introduce new security and privacy risks by revealing personal user information such as list of users' social contacts and profile information [16, 17] .

The primary reason of these issues is the complete control of a centralized social network provider, for example Facebook or Google+, on a large collection of user data. Users trust the provider with their data. However, the provider can use the user data any way it wants. Therefore, researchers have proposed several designs to shift the access enforcement point from the OSN provider to the end users. Unfortunately, not all these proposed schemes provide adequate level of privacy. Several designs [18–21] focus on cryptography-based access control while keeping the centralized architecture of the OSN provider. Centralized OSN providers are interested in user data

since they can sell it with advertisers who can send targeted advertisements based on this data.

On the other hand, users interested in using the OSN functionalities must trust the provider hoping that it will not misuse their data. However, concerned users share limited information or in some cases restrict themselves from participating in any OSN. Hence, a decentralized architecture is more desirable for enhancing security and privacy in OSNs. Researchers have worked on designs [22–26] that primarily focus on decentralization of the OSN rather than security and privacy. A recent work [27] combines both cryptography and distributed storage service, but falls short supporting access revocation, which is a required feature to support the type of policy expected by social network users. None of these proposed works allow users to get an idea about who accesses their data while providing adequate level of security and privacy.

Apart from socialization, social networking is being used to support patients who have similar symptoms and want to get real world experience [1]. It provides a platform where patients share different types of medical information, and this information is shared with third parties [28]. Medical data is always considered sensitive, and hence, this kind of OSN should be secure and private. In future, when this type of social networking will be popular and widely used to support broader functionality, such as health information exchange, patients will need more control on their data rather than trusting the social network provider.

Decentralization will enable patient-centric medical social networking where patients are in control of their data. They can share their data with whoever they want to get service without revealing this information to the social net-

[1] http://www.patientslikeme.com

work provider. A secure health information exchange system will be possible so that users get better service from the healthcare providers. We envision a social networking system with enhanced security and privacy features that will be used not only for better socialization but also improved healthcare system.

1.1 Thesis Statement

In this thesis we investigate, analyze, and propose techniques to enhance security and privacy in OSN. We focus on decentralization of the social networking architecture and cryptographic techniques. We believe that providing security and privacy in OSNs is challenging. We also understand that it may not be achievable completely by hiding every piece of information since that may interfere with the services provided by the OSN. Our work can be summarized through the following research statement.

> *Security and privacy of online social networks can be enhanced by shifting the control over user data from the centralized provider to the end users through a decentralized architecture and advanced cryptographic mechanisms.*

1.2 Challenges

Decentralization of the social networking architecture shifts the control over data to the end user and hence enhances security and privacy. However, designing a secure and private decentralized social network is challenging.

- Distributing data requires careful consideration. In order to provide confidentiality and integrity, researchers have proposed to encrypt data.

However, as research shows, existing cryptographic techniques are not adequate to support the flexible policy expected by social network users [27].

- Encryption only may not be enough to provide privacy. Appropriate measures should be taken to avoid any kind of information leak, for example, inferring social relationships.

- Since there is no single platform to store and serve user data, ensuring data availability gets challenging.

- Social networking data is of different types and may have multiple ownership – for example, a comment on a status post belongs to both the status poster and the commenter. Therefore, designing a data structure that supports the complex security and privacy expectation on social networking data is challenging.

- Finally, deployment of a decentralized OSN is the most challenging task since it has to provide better security and privacy than existing OSNs while allowing efficient service. However, our focus in this thesis is on enhancing the security and privacy aspects, and we leave wide scale deployment as a future work.

1.3 Requirements and Properties

1.3.1 Security and Privacy Requirements

We can outline a number of security and privacy requirements of a decentralized social network:

Confidentiality: Preserving the confidentiality of user content is a key requirement for a decentralized OSN. Content should be accessible to only those who are explicitly authorized by the content owner. Furthermore, nodes hosting such data may themselves not be authorized to read the data.

Integrity: We must also ensure the integrity of the data so that OSN users can be certain that the content posted by their friends is authentic. This property is important in a peer-to-peer network since storage nodes are untrusted and may try to perform unauthorized updates to the stored data.

Availability: User content should remain available until it is explicitly deleted by its owner, even if the owner is offline, and despite potential malicious attempts to destroy the data. Readers should also be able to retrieve the most recent version of a content object rather than past ones.

Explicit Owner-specified Access Control: Policies controlling who may view, modify, or comment on content are defined by its owner and cannot be changed without the owner's authorization. However, when a content has multiple owners, the more restrictive policy should be enforced.

Relationship Privacy: User's social graph is considered sensitive [29]. Relationships between users should remain hidden from third parties that may have no relationship with the data owner and are therefore untrusted, such as storage nodes.

Auditing: Users should be be able to get information about when and by whom their data is used for better knowledge about privacy. Stealing or sharing a digital data does not leave any trace. Informative auditing should be enabled at every access of and action on information at each level, which can be used in future to detect potential misuse of data.

Authentication: Information accessors should authenticate themselves at each access of data. Identity of the accessors is a crucial part in tracing

back misuse of data. However, authentication should not reveal any information to inappropriate parties.

1.3.2 Threat Model

We assume that the participants in the decentralized OSN may be malicious (or compromised) and therefore should not be trusted with the confidentiality and integrity of data and relationship information. Moreover, the malicious entities are considered to be Byzantine, and can launch both active and passive attacks. Distributed systems are vulnerable to the problem of Sybil attacks, where a single entity can obtain multiple identities in the system and violate security properties. We assume the existence of mechanisms to defend against the Sybil attack [30, 31]. We consider that up to 25% of the nodes in the system can be malicious, since, beyond that, existing mechanisms [30] are not able to securely route in distributed hash tables, which is a necessary prerequisite to provide both integrity and availability guarantees.

In peer-to-peer networks, malicious nodes may attempt to launch denial of service (DoS) attacks against honest peers, by overwhelming their network, computational, or storage resources. We assume existence of defenses against such DoS attacks [32, 33]. In this thesis we focus on architecture design and cryptographic mechanisms to protect the security of stored data, rather than on routing-based attacks or DoS attacks, which can be addressed by existing mechanisms, and which we will therefore not discuss here.

1.4 Proposed Approaches

We propose three methods to enhance security and privacy in OSNs. These methods should not be considered mutually exclusive, but combined together

they build up a secure and privacy-aware social network.

Our first approach named EASiER [34,35] is a cryptographic access control scheme for social networking. Researchers have proposed *Attribute-based Encryption* (ABE) to ensure data confidentiality in social networking because of its expressiveness [27]. Unlike traditional cryptography, in ABE, data is encrypted with a plaintext policy defined over a set of attributes, for example *'(friend and family) or coworker'*. Secret keys that satisfy this policy will be able to decrypt this data. However, ABE schemes can be expensive to provide perfect forward secrecy when the system requires frequent revocation as they are burdened with redistributing individual secret keys. For example, in social networking systems, users may encrypt data for their *close friends*, but cease the relationship with a specific friend at a later time. In absence of an efficient revocation scheme, they will have to re-key every remaining *close friend* to protect data, which is an expensive task if the list is large. We present an efficient revocation scheme in ABE by introducing a *minimally trusted proxy*. Though we present it in the context of a social network, it can be adopted by any system that uses ABE to protect its data.

We propose DECENT [36, 37], a decentralized design and architecture for OSN with an emphasis on confidentiality, integrity, and availability. We distribute user data and get rid of the centralized OSN provider. The underlying storage system is a distributed hash table (DHT) that structurally stores and retrieves private user data. We borrow the concept of object oriented design to represent different types of social networking data such as user profiles, wall posts, pictures, and messages. The data owner creates an object with the data as an object element, and stores the object in the DHT. This operation gives the user a reference to the object in the DHT. Since data is distributed, each object along with its reference is encrypted to provide confidentiality.

The objects and their references as well are encrypted using EASiER. This architecture shifts the access control point from the provider to the users by combining advanced cryptographic mechanisms and decentralization.

For sensitive data, for example, interactions in the social network, it is important to avoid revealing undue information to the storage provider. Despite encryption, certain information, such as access patterns, are still available to the storage provider and it can tell, for example, who accessed our data, or who we interact with socially. Cryptographic access control makes it difficult to implement important security protections, such as maintaining an audit log of accesses. In the third approach, we allow a data owner to receive a log of data accesses, while hiding this information from the storage provider. We call this approach auditable anonymity. To address the auditability issue, we extend anonymous credentials with revocable anonymity [38] that allows users to prove in zero knowledge that they are authorized to access a service or a piece of data without revealing the contents of the credential, and, importantly, their identity through a chain of trust, a concept similar to anonymous webs of trust [39]. Each access, however, generates an encrypted record that can be used by a special anonymity revocation service to learn who was behind such an access. In our architecture, this service is run by the data owner, who is able to decrypt the audit log records and get an access log.

1.5 Summary of Contribution

Our contribution can be summarized as follows:

- We construct and implement a cryptographic scheme that supports efficient revocation in attribute-based encryption. The code is available

at `https://bitbucket.org/hatswitch/easier`.

- We construct and implement a cryptographic scheme that allows anonymous access on outsourced user data while creating a private access log for the data owner hidden from the storage provider.

- We also design and implement a decentralized social networking architecture that focuses on data confidentiality, integrity, and availability.

We adopt and use the cryptographic schemes in the context of enhancing security and privacy of online social networking. However, these approaches are generalized and hence, any outsourced storage will be able to adopt these techniques to provide security, privacy, and anonymity to protect their data.

1.6 Organization

In the rest of this thesis, we give a literature review in Chapter 2. We describe the three approaches that we propose to enhance security and privacy in OSNs in Chapters 3, 4, and 5 respectively, and briefly summarize in Chapter 6 how these approaches are combined for a secure and privacy-aware social network. Finally, we conclude in Chapter 7.

2 LITERATURE REVIEW

In this chapter we describe several existing works that try to address the security and privacy issues of OSNs. At first we present a summary of the existing works and why they fall short, and next we describe them in detail.

2.1 Summary

Researcher have been working on different approaches to address the security and privacy problems of online social networking. These works broadly can be categorized based on the architecture of the social network provider (centralized or decentralized) and the type of mechanism used to provide security and privacy of stored data (cryptography, information flow, or trust). Most of these works focus on one aspect rather than the overall design with emphasis on security and privacy.

Some designs require users to trust the centralized provider with their data, but focus on how information is shared with third party applications [40]. Some focus on auditing without emphasis on security and privacy [41]. Several works use different types of cryptographic techniques as their protection mechanism [18–20, 42–45]. In these approaches, the social network provider may not have enough incentive to store encrypted data as it does not have direct access to user data, or may infer a user's social graph as it performs key management or mediates access to users' data.

Decentralization of the architecture allows more control over data to the

users as it removes the concept of trusting a centralized provider [22–27, 46]. In decentralized designs, data is stored with trusted or untrusted storage nodes, which can be third party storage servers or other users in the social network. In some designs users trust their social contacts to enforce their privacy policies, some rely on a few supernodes for availability and access control, and some trust the storage nodes with their data. A few designs use cryptographic techniques for data protection.

However, several problems still exist. Existing designs do not take into consideration the different types of data in OSNs, they do not consider complex data ownership and flexible policies, and do not have option to track data access while protecting data. Besides, when encryption is used to protect user data, existing schemes fall short to cryptographically represent privacy policies expected by OSN users. We for the first time, present a comprehensive design for a decentralized OSN, and construct novel cryptographic schemes to support appropriate access control in social networking and anonymous auditing on protected user data.

2.2 Centralized Social Networking Architecture

Singh et al. propose the xBook framework for building privacy preserving social network applications [40]. xBook uses information flow models to control what untrusted applications can do with the information they receive. Their design retains the functionality offered by existing online social networks. xBook provides enforcement for both user-user access control for information flow within a single application, as well as for information sharing with entities outside xBook. Social network applications are re-designed to have access to all the data that they require, but this data is not allowed to be

passed on to an external entity unless approved by the user. However, xBook does not perform encryption for data confidentiality and hence the security and privacy risks persist.

Cristofaro et al. propose Hummingbird, a privacy enhanced version of Twitter type social network [42]. In this architecture, users encrypt their tweets and can follow hash-tags without revealing this information to unauthorized parties. Hummingbird is a centralized social networking design and the provider does not have access to the tweet contents because of encryption, which may not motivate it to store user data.

Lucas and Borisov propose flyByNight, a Facebook application designed to mitigate privacy risks in social networks [19]. flyByNight users encrypt sensitive messages using javascript on the client side and send the ciphertext to some intended party, i.e., Facebook friends, who can then decrypt the data. The architecture ensures that transferred sensitive data cannot be viewed by the Facebook servers in an unencrypted form. However, the utility of flyByNight is limited to preserving the privacy of messages intended for social network friends, i.e., email type communication, and thus, it does not provide complete privacy. For example, the application server knows a user's friendlist on facebook. flyByNight is also vulnerable to active attacks by the OSN provider, since the OSN interface is used for key management.

Guha et al. propose to improve user privacy while still preserving the functionality of existing online social network providers [18]. Their architecture is called *none of your business* (NOYB), in which encryption is used to hide the data from the untrusted social network provider. The key feature of their architecture is a general cipher and encoding scheme that preserves the semantic properties of data such that it can be processed by the social network provider *oblivious* to encryption. A users private information is partitioned

into *atoms*, and NOYB encrypts a users atom by substituting it with the atom of another user. Thus the OSN can operate on ciphered data, but only the authorized users can decrypt the result.

Frikken describes a key allocation scheme for OSNs [20]. In this work, relationship between two users depends on their distance. A social contact at a distance less than d can access any information encrypted by the user for her contacts at distance d. Each user defines the maximum depth L of her social relationships, generates keys for each distance, encrypts data for her contacts at the desired distances, and makes the keys available to the appropriate parties. However, to establish different relationships such as friend, family, etc. a user defines $L + 1$ keys for each relationship, making it infeasible.

Luo et al. propose FaceCloak where users store random and fake data on Facebook and the actual data in encrypted form on FaceCloak server [43]. Fake data from Facebook is used as an index to retrieve actual data from FaceCloak on the client side. This design introduces redundant data storage at two different servers. Beato et al. propose Scramble, which uses symmetric key encryption to provide data confidentiality and integrity [44]. However, Scramble also creates an access control list of users who are actually allowed to see a piece of data by encrypting the symmetric key to each of them separately. Hence, the approach is not expressive.

Anderson et al. propose a client-server architecture for providing social network privacy [21]. In their design, the server is a very simple untrusted social network provider which serves as a data container, while a complex client side architecture performs the access control. The server only provides availability and client is responsible for data confidentiality and integrity. Besides content data, the architecture is also able to protect the social graph

information.

Feldman et al. propose a framework called Frientegrity that primarily focuses on avoiding equivocation by the untrusted centralized social network provider [45]. Users store their encrypted data with the provider, and verify the data integrity through a history of operations on the data and collaboration with other users. Although Frientegrity provides several desired features of a security and privacy-aware social network, the provider is still centralized, and therefore, may not have enough incentive to store user data. Besides, access revocation is logarithmic with the number of friends a user has. Also, provider can deduce an anonymous graph of relations between users.

2.3 Decentralized Social Networking Architecture

Shakimov et al. consider a decentralized OSN as a privacy preserving alternative to the current centralized architectures [22]. The basic idea of their decentralized OSN is that users store their data in a Virtual Independent Server (VIS) owned by themselves. These VISs form an overlay network per OSN. The authors consider different types of decentralized OSNs, depending on where the VISs reside: a) cloud based b) desktop based c) hybrid, and compare their privacy, costs and availability.

PeerSon, LotusNet, and Safebook [23–25], three decentralized designs for social networking benefit from DHTs in their architecture. PeerSon and Safebook suggest access control through encryption, but they fall short to provide fine-grained policies compared to ABE-based access control. Safebook is based on a peer-to-peer overlay network named *Matryoshka*. The end-to-end privacy in Matryoshka is provided by leveraging existing hop-by-hop

trust. In all of these schemes overhead of key revocation affects performance. LotusNet [23] is a decentralized OSN based on a DHT called Likir [47], which in turn is based on the base DHT Kademlia [48]. To provide access control, Likir uses signed grants to specify permissions. Though it is based on a DHT with untrusted nodes, it does not support any kind of encryption.

Diaspora is a decentralized client-server architecture for OSNs where each user is considered to own a *seed* [26]. A seed is a web server, contains user information, and can reside anywhere including the user's machine and the diaspora server. Diaspora defines three types of security level including high, low, and no security at all. However, Diaspora is not based on a DHT and uses symmetric key encryption for high security data. The access policies are not fine-grained or expressive, and revocation is still a problem in Diaspora.

SCOPE is a distributed data management system for specialized P2P social networks [46]. Clients connect to and store data on a group of super-nodes with higher computation and storage capacity. However, clients do not participate in the DHT; only the super-nodes run the DHT code. Clients connect to super-nodes and rely on them for sharing and access control on their data.

Persona is a privacy-aware OSN architecture based on ABE [27]. Persona uses a decentralized persistent storage medium, and puts policy decisions in the hands of users by allowing them to choose with whom they store their information. Users are empowered to create groups and are also able to mediate admission control in a group. Access to personal data is controlled by encrypting to groups. Restricting data to specific groups allows users to have fine-grained control over access policy. Users define relationships with their social contacts by assigning attributes to them. For example, Alice might assign the attributes *friend* and *co-worker* to Bob. Data is encrypted under a policy defined over a set of attributes using potentially complex logical

formulas. Users can only decrypt the data if they have enough attributes to satisfy the policy. Though Persona introduces a desired property for OSN by performing ABE, it does not scalably support dynamic groups. The major problem is key revocation. Whenever someone leaves a group, Alice needs to issue the remaining members new keys, and the entire data set may need re-encryption. This creates high overhead if group membership changes frequently.

3 EASIER

Encryption allows users to directly control access to their data by distributing keys to the intended parties. Fine-grained access control is a key challenge in this space; for example, Facebook and Livejournal have rolled out mechanisms to specify access control policies for each post, as the data items are usually destined for a subset of friends, or groups. Persona [27] is a state-of-the-art design that proposes the use of *Attribute-based Encryption* (ABE) [49] to enable fine-grained access control. A user can create groups by assigning different attributes and keys to her social contacts, and then encrypt data such that only particular users having the desired set of attributes can decrypt it.

However, user attributes may change over time and access revocation becomes necessary. This could be because of change in location, work environment, or the nature or strength of the relationship with a contact [50]. Recent studies have shown that the user interaction graph is much less dense than friendship graph, indicating that users interact most frequently with a small group of friends, further validating the need for fine grained access control [51]. Moreover, the churn rate for the interaction graph has been shown to be quite high, motivating the need for access control mechanisms to support *dynamic groups*. Persona and similar designs introduce significant overhead for group membership changes, especially when a contact is removed from a group: all other members of the group must receive a new key; additionally, all existing data items destined for that group must be re-

encrypted. This does not scale when group sizes are large and group churn rate is high.

Our first approach EASiER is a cryptographic scheme that enables users to set fine-grained access control policies even for dynamic groups. To handle group churn, we leverage ideas from the field of efficient revocation in group communications systems [52] and apply them to the ABE setting. Our design makes use of a minimally trusted proxy that handles revoked users and attributes. A user who revokes a contact or an attribute need not issue new keys to the rest of the group, nor re-encrypt data. We believe this feature is key for access control in OSNs, and would also be useful in any other context where ABE is used together with highly dynamic group membership.

The proxy cannot decrypt by itself, and even if it were compromised, it cannot allow previously revoked users to decrypt either (unless the compromise crosses a proxy re-key operation). Therefore, a centralized OSN provider can act as a proxy without introducing significant privacy risks. The only assumption we hold is that the proxy is updated with a new key each time a revocation takes place and that it discards the old one. We provide a proof of the security of our construction, and present performance analysis that demonstrate the efficiency of the scheme.

3.1 Background

3.1.1 Attribute-based Encryption

With standard public-key cryptography, it is necessary to explicitly enumerate all of the users who may decrypt each data item. While it is possible to issue group keys, this creates complex key management issues and several

problems still remain. Attribute-based encryption (ABE) [49,53,54] provides a better solution for defining fine-grained access control to groups of people. It provides an alternative approach to data protection, where the ability to decrypt data items is controlled by a policy specified in terms of attributes. For example, in Ciphertext-Policy ABE (CP-ABE) [49] (one variant of ABE), the encryptor uses a policy described over attributes to encrypt a piece of data; different secret keys are issued for different sets of attributes, and a key that has enough attributes to satisfy the policy decrypts the ciphertext.

For example, Alice encrypts a file with the policy "*Professor OR (Teaching Assistant AND Security)*", which means either a Professor, or a Teaching Assistant in Security can decrypt this file. An example key with attributes "Teaching Assistant, Security" that Alice issues can decrypt this file since it satisfies the policy, but a key with just the "Teaching Assistant" attribute cannot. This variant is called *ciphertext-policy* ABE; *key-policy* ABE reverses the process, with attributes assigned to data items and policies to keys.

ABE explicitly prevents collusion between users: if Bob is a teaching assistant in databases, and Carol is a research assistant in security, they cannot combine their attributes together to satisfy the above policy if neither of them satisfies it individually. Finally, ABE provides public-key functionality, allowing, for example, Bob to encrypt a message with Alice's public key to be decrypted by the set of secret keys issued by Alice.

Various applications can benefit from the flexibility and expressiveness of ABE, but require support for frequent revocation. ABE falls short in such areas and when immediate revocation of access is required. Researchers have proposed revocation by attaching an expiry date to the keys [49, 54], which requires re-keying non-revoked users and re-encrypting old ciphertexts. They also introduce delay in revocation.

We can formally define a CP-ABE scheme with revocation by six algorithms:

- Setup. This algorithm takes security parameters and generates a public key PK and master secret key MK.

- Encrypt(PK, M, P). This algorithm takes the public key PK, a message M, and a policy P and generates a ciphertext CT encrypted with P.

- Keygen(MK, S). This algorithm uses the master secret key MK to generate a secret attribute key SK using the attributes in the set S.

- ProxyRekey(PK, MK, U**)**. This algorithm uses the public key PK and master secret key MK to generate a proxy key PXK using the revoked users in the set U.

- Convert(PXK, CT, u**)**. This algorithm, run by the proxy, converts the ciphertext components in the set CT into a decryptable form CT' for the user u using the proxy key PXK.

- Decrypt(CT', SK). This algorithm decrypts a converted ciphertext CT' to plaintext M as long as the set of attributes S in SK satisfies the policy P that was used to generate CT from M. (The policy is implicitly encoded in CT.)

- Delegate$(SK, \tilde{S})$. The delegate algorithm in single authority setting takes as input a secret key SK for some set of attributes S and a set $\tilde{S} \subseteq S$. It outputs a secret key $\tilde{SK}$ for the set of attributes $\tilde{S}$. The algorithm re-randomizes the secret key and allows further delegation.

- DelegatedDecrypt$(CT', \tilde{SK})$. This algorithm decrypts a converted ciphertext CT to plaintext M, as long as the set of attributes $\tilde{S}$ in $\tilde{SK}$ satisfies the policy P that was used to generate CT from M.

We also consider a multi-authority delegation scenario. In this case, Alice runs a key authority that issues a secret key to Bob, who himself runs a

separate key authority that can delegate the key to Carol. Alice and Bob each run separate proxies and Carol may decrypt only if Alice has not revoked Bob and Bob has not revoked Carol.

- MA–Delegate$(SK, \tilde{S}, \tilde{MK})$**.** This allows Bob to delegate his secret key SK, issued by Alice for the set of attributes S, using his own master secret key $\tilde{MK}$, producing a delegated key $\tilde{SK}$, with attributes $\tilde{S} \subset S$

- MA–DelegatedDecrypt($CT'', \tilde{SK}$**).** This allows the owner of a delegated key to decrypt a ciphertext CT'' that was obtained by first running CONVERT using Alice's proxy with key PXK, and then converting it again using Bob's proxy and key $P\tilde{X}K$. The decryption succeeds if $\tilde{S}$ satisfies the policy P used to encrypt the original ciphertext.

3.1.2 Revocation Scheme

To support practical revocation in EASiER, we adopt a scheme developed by Naor and Pinkas [52] that allows efficient revocation of users. The scheme uses Shamir secret sharing [55] to create shares of a secret key, where $t + 1$ shares are necessary for reconstruction, and gives one to each user. During normal operation, the sender broadcasts t random shares, which lets any user to reconstruct the secret key by combining the shares with his or her own. To revoke up to t users, the sender broadcasts their shares instead of random ones. Any non-revoked user still has $t+1$ distinct shares and can reconstruct the private key, whereas the revoked users do not have enough information even if they all collude.

3.1.3 Cryptography Basics

Bilinear Pairing:

Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be multiplicative cyclic groups of prime order p, and e a map ($\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$). Let g_1 and g_2 be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively ($\mathbb{G}_i = \langle g_i \rangle$). If $\forall u \in \mathbb{G}_1, v \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$, $e(u^a, v^b) = e(u, v)^{ab}$ and $e(g_1, g_2) \neq 1$, then e is called a bilinear pairing. If $\mathbb{G}_1 = \mathbb{G}_2$, it is called a symmetric pairing, otherwise the pairing is asymmetric.

Secret Sharing:

In Shamir's secret sharing scheme [55], a secret s in some field F is shared among n parties by creating a random polynomial $P \in F[x]$ of degree t such that $P(0) = s$. The i-th party gets the share $(i, P(i))$. Given any $t+1$ shares $P(x_0), \ldots, P(x_t)$, it is possible to recover $P(0)$ using Lagrange interpolation:

$$P(0) = \sum_{i=0}^{t} \lambda_i P(x_i), \quad \text{where } \lambda_i = \prod_{j \neq i} \frac{x_j}{(x_j - x_i)}$$

3.2 Architecture

Unlike traditional OSNs, social relationships in EASiER are of various types. Users define relationships by assigning arbitrary set of attributes to their contacts, act as their own key authorities, generate different secret keys for the assigned attributes, and encrypt pieces of data such as profile information, wall posts, etc. with attribute policies. An encrypted item can be decrypted only by the contacts with keys that have enough attributes to satisfy the policy attached to the data. For instance, in Figure 3.1, user Alice defines the attributes (*friend, colleague, neighbor*), generates the keys k_1 ('colleague'), k_2 ('friend, neighbor'), and k_3 ('colleague, neighbor') for u_1, u_2, and u_3 re-

spectively. Therefore, u_1 and u_3 are in the colleague group, u_2 and u_3 are in the 'neighbor' group, and u_2 is in the 'friend' group of Alice. She encrypts her data with the policy *'colleague or (friend and neighbor)'*.

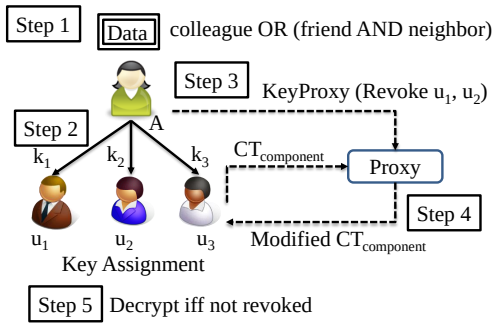


Figure 3.1: EASiER Architecture

Revocation is a key functionality for OSNs, as relationships can and do change over time: Alice may want to cease her relationship with u_1 and u_2, and revoke the corresponding keys, which should prevent them from viewing her data encrypted with any policy that their keys satisfy. Besides, she may want to revoke the attribute 'neighbor' from k_3 assigned to u_3 and effect a corresponding change in access control. EASiER makes use of a proxy that is involved in every decryption. In summary, decryption keys in EASiER are blinded in a way specific to each user's identity; to decrypt a data item, a user with the appropriate key must contact the proxy to transform the ciphertext partially so that it is compatible with the blinded key. When Alice wishes to revoke an attribute, she updates the proxy key in such a way that this transformation is impossible for the revoked users.

The proxy itself is minimally trusted and cannot decrypt the data; it also cannot restore access for a previously revoked user even if it is compromised, and so it may be implemented by a centralized service with minimal risk. Upon each revocation, Alice rekeys her proxy with the latest revocation in-

formation. A new proxy key, created each time a revocation takes place, prevents revoked users from colluding with the proxy or with each other to get the data. We argue that this is a desirable property: currently trusted contacts are not likely to crawl the entire set of social network data and store it for later use, but former friends or colleagues might try to abuse their former status by accessing past data. Figure 3.1 summarizes the whole architecture.

3.3 Cryptographic Construction

3.3.1 Key Revocation

Intuition:

The master key MK contains a polynomial P of degree t. Each user u gets a random share $P(u)$ of $P(0)$ in her key. The proxy key consists of t such random shares and is used to convert a part of the ciphertext for decryption. Whenever an access is revoked from a key SK, its share becomes a part of the proxy key, and hence the converted ciphertext. Therefore, the revoked user does not have $(t+1)$ distinct points to unblind her key and the ciphertext. However, non-revoked users always have $(t+1)$ distinct points and hence decrypt successfully.

When no one is revoked, the proxy key consists of t random $P(x)$ points. Since the revocation is based on polynomial secret sharing with a polynomial of degree t, the scheme is limited to a maximum of t key revocations. The total number of users in the system, however, is not limited.

- Setup: The setup algorithm randomly generates a polynomial P of degree t over $\mathbb{Z}_p$, where $p = |\mathbb{G}_1|$, sets the secret $P(0)$, and randomly chooses

$\alpha, \beta \in \mathbb{Z}_p$. The public key PK and master secret key MK are generated as follows. This is directly adapted from BSW CP-ABE, modifying it to use asymmetric pairings and adding the polynomial component to the secret key.

$$PK = \left(\mathbb{G}_1, \mathbb{G}_2, g_1, g_2, h = g_1^{\beta}, e(g_1, g_2)^{\alpha}\right), MK = (\beta, g_2^{\alpha}, \mathbf{P})$$

- Encrypt(PK, M, τ): A policy is represented as an access tree structure τ with the attributes at leaves and threshold k-of-n gates at intermediate nodes. Each node x is associated with a polynomial q_x of degree d_x ($d_x = k - 1$ for that node). The random secret s ($s \in \mathbb{Z}_p$) to blind the data M is associated with the polynomial at the root R of the tree; i.e., $q_R(0) = s$. For all other nodes, $q_x(0) = q_{parent(x)}(index(x))$. $index(x)$ returns a number between 1 and the number of children of $parent(x)$. Y is the set of leaf nodes in τ.

Other than the asymmetric groups, the encryption algorithm works exactly the same as in BSW CP-ABE. $H : \{0,1\}^* \rightarrow \mathbb{G}_2$ is a hash function (modeled as random oracle) that maps string attributes to random elements of $\mathbb{G}_2$, and $h_y = \log_{g_2} H(att(y))$, used for notational convenience only, as it is assumed infeasible to compute.

$$CT = (\tau, \tilde{C} = Me(g_1, g_2)^{\alpha s}, C = h^s = g_1^{\beta s},$$
$$\forall y \in Y : C_y = g_1^{q_y(0)},\ C'_y = H(att(y))^{q_y(0)} = g_2^{h_y q_y(0)})$$

- KeyGen(MK, S): The key generation algorithm outputs the secret key corresponding to the set of attributes S for the user u_k, blinded by $P(0)$. u_k and D''_j are the extra components generated by this algorithm compared to the same algorithm in BSW CP-ABE. The component D''_j contains user

information u_k (a point on P). u_k does not need to be secret since P is secret.

$$SK = (D, \forall j \in S : \langle D_j, D'_j, \mathbf{D''_j} \rangle, u_k),$$
$$D = g_2^{(\alpha+r)/\beta}, D_j = g_2^r H(j)^{r_j \mathbf{P(0)}}, D'_j = g_1^{r_j}, \mathbf{D''_j} = (\mathbf{D'_j})^{\mathbf{P(u_k)}} = \mathbf{g_1^{r_j P(u_k)}}$$

- ProxyRekey($PK, MK, \{u_1, \ldots, u_r\}$): This algorithm takes a list (possibly empty) of revoked users, with $0 \leq r \leq t$ and evaluates the corresponding $P(u_i)$ using MK. In case of fewer than t revocations, it generates random points x (other than the actual user identities) and calculates $P(x)$. The secret proxy key PXK is set as follows:

$$\begin{aligned} PXK = &\{u_i, P(u_i)\}, 1 \leq i \leq r \\ &\{x_i, P(x_i)\}, r < i \leq t, \qquad \text{for random } x_i \end{aligned}$$

- Convert(PXK, CT): This algorithm, run by the proxy, takes as input the proxy key PXK, the ciphertext CT, and the decryptor's identity u_k to calculate C''_y as follows:

$$\forall i, j \in \{1, \ldots, t\},\ k \notin \{1, \ldots, t\}, \lambda_i = \frac{u_k}{u_k - u_i} \cdot \prod_{j \neq i} \frac{u_j}{(u_j - u_i)},$$
$$\forall y \in Y : C''_y = (C'_y)^{\sum_{i=1}^{t} \lambda_i P(u_i)} = g_2^{h_y q_y(0) \sum_{i=1}^{t} \lambda_i P(u_i)}$$

Since SK is blinded by $P(0)$, the user needs C''_y in addition to C_y and C'_y for decryption. The algorithm also calculates λ_k for the decryptor u_k, so $CT' = \{CT, \{\forall y \in Y : C_y\}, \lambda_k\}$. Note that in practice, only the C'_y components of the ciphertext need to be passed to the proxy.

- Decrypt(CT', SK): The decryption steps involve one extra pairing than BSW CP-ABE at each leaf node of the policy. For each leaf node x where

$i = att(x)$ and $C''_x \in C''$, if $i \in S$, (S is the set of attributes for which SK is issued) then,

$$\begin{aligned}
DecryptNode(CT, SK, x) &= \frac{e(C_x, D_i)}{e(D''_i, C'_x)^{\lambda_k} e(D'_i, C''_x)} \\
&= \frac{e(g_1, g_2)^{rq_x(0)+h_i r_i P(0) q_x(0)}}{e(g_1, g_2)^{r_i h_i q_x(0) \lambda_k P(u_k)} e(g_1, g_2)^{r_i h_i q_x(0) \sum_{j=1}^{t} \lambda_j P(u_j)}} \\
&= \frac{e(g_1, g_2)^{rq_x(0)+h_i r_i P(0) q_x(0)}}{e(g_1, g_2)^{r_i h_i q_x(0) (\sum_{j=1}^{t} \lambda_j P(u_j) + \lambda_k P(u_k))}} \\
&= \frac{e(g_1, g_2)^{rq_x(0)+h_i r_i P(0) q_x(0)}}{e(g_1, g_2)^{r_i h_i q_x(0) P(0)}}, k \notin \{1, 2, \ldots, t\} \\
&= e(g_1, g_2)^{rq_x(0)}
\end{aligned}$$

Otherwise *DecryptNode* returns $\perp$. The rest of the decryption proceeds as BSW CP-ABE.

Explanation of Asymmetric Group:

We use different groups for C'_i and D'_i. The user gets C'_i converted to $C'''_i = C'^a_i$ ($a = \sum_{j=1}^{t} \lambda_j P(u_j)$, explained in the CONVERT algorithm) by the proxy. However, if both C'_i and D'_i belong to the same group, and the user gives the proxy D'_i instead of C'_i, she will get $(D'_i)^a = g^{ar_i}$. She will also get λ_k, which she can use to get $D''^{\lambda_k}_i = g^{\lambda_k P(u_k) r_i}$. Multiplying these two, she gets $g^{r_i(a+\lambda_k P(u_k))} = g^{r_i P(0)}$. She can use this last value to decrypt any ciphertext without using the proxy, so the revocation is no longer effective. Therefore, we use asymmetric pairing, where C'_i and D'_i are in different groups and mapping of D'_i into the C'_i group is not possible, also known as External Diffie Hellman Assumption (XDH).

3.3.2 Delegation of Access

We design delegation of attributes in EASiER in two settings depending on whether the secret keys are issued by a) single key authority, or b) multiple key authorities.

- Delegate$(SK, \tilde{S})$. The delegation algorithm takes in a secret key SK issued for a set of attributes S to user u_k, and generates a delegated key $\tilde{SK}$ for the subset of attributes $\tilde{S} \subseteq S$. As long as u_k is not revoked, the delegated key can be used for decryption. An extra public parameter $f = g_2^{1/\beta}$ is introduced in the public key PK. Some components in SK are re-randomized and a new component D_j''' is introduced. Let random $\tilde{r} \in Z_p$, and random $\forall_j \in \tilde{S}, \tilde{r}_j \in Z_p$. $\tilde{SK}$ is set as follows.

$$\tilde{SK} = (\tilde{D}, \forall j \in \tilde{S} : \langle \tilde{D}_j, D_j', D_j'', D_j''' \rangle),$$
$$\tilde{D} = (D) f^{\tilde{r}} = g_2^{(\alpha + r + \tilde{r})/\beta}, \tilde{D}_j = (D_j) g_2^{\tilde{r}} H(j)^{\tilde{r}_j}, D_j''' = g_1^{\tilde{r}_j}$$

- DelegatedDecrypt$(CT', \tilde{SK})$. Decryption proceeds as follows. For

each leaf node x and $C''_x \in C''$:

$$
\begin{aligned}
&\mathit{DecryptNode}(CT, \tilde{SK}, x) \\
&= \frac{e(C_x, \tilde{D}_i)}{e(D_i''^{\lambda_k} \cdot D_i''', C'_x) e(D'_i, C''_x)} \\
&= \frac{e(g_1, g_2)^{q_x(0)(r+\tilde{r}+h_i r_i P(0)+h_i \tilde{r}_i)}}{e(g_1, g_2)^{(r_i \lambda_k P(u_k)+\tilde{r}_i) h_i q_x(0)}} \cdot \frac{1}{e(g_1, g_2)^{r_i h_i q_x(0) \sum_{j=1}^{t} \lambda_j P(u_j)}} \\
&= \frac{e(g_1, g_2)^{(r+\tilde{r}) q_x(0)+h_i r_i P(0) q_x(0)+h_i \tilde{r}_i q_x(0)}}{e(g_1, g_2)^{r_i h_i q_x(0) \lambda_k P(u_k)+h_i \tilde{r}_i q_x(0)}} \cdot \frac{1}{e(g_1, g_2)^{r_i h_i q_x(0) \sum_{j=1}^{t} \lambda_j P(u_j)}} \\
&= \frac{e(g_1, g_2)^{(r+\tilde{r}) q_x(0)+h_i r_i P(0) q_x(0)+h_i \tilde{r}_i q_x(0)}}{e(g_1, g_2)^{r_i h_i q_x(0) (\sum_{j=1}^{t} \lambda_j P(u_j)+\lambda_k P(u_k))+h_i \tilde{r}_i q_x(0)}} \\
&= \frac{e(g_1, g_2)^{(r+\tilde{r}) q_x(0)+h_i r_i P(0) q_x(0)+h_i \tilde{r}_i q_x(0)}}{e(g_1, g_2)^{r_i h_i q_x(0) P(0)+h_i \tilde{r}_i q_x(0)}}, k \notin \{1, 2, \ldots, t\} \\
&= e(g_1, g_2)^{(r+\tilde{r}) q_x(0)}
\end{aligned}
$$

Let $A = e(g_1, g_2)^{(r+\tilde{r}) q_R(0)} = e(g_1, g_2)^{(r+\tilde{r}) s}$ for the root node. The rest of the decryption proceeds as follows,

$$
\frac{\tilde{C}}{\frac{e(C, \tilde{D})}{A}} = M e(g_1, g_2)^{\alpha s} \frac{e(g_1, g_2)^{(r+\tilde{r}) s}}{e(g_1, g_2)^{\alpha s+r s+\tilde{r} s}} = M
$$

- MA-Delegate($SK, \tilde{S}, \tilde{MK}$). Recall that in this case, Bob is in possession of a secret key SK issued by Alice, and is simultaneously in charge of a separate key authority with master key $\tilde{MK}$. In this case, Bob creates a new identity u' for another user Carol and re-randomizes the delegated key: $\tilde{S} \subseteq S$.

$$
\begin{gathered}
\tilde{SK} = (D, \forall j \in \tilde{S} : \langle D_j, D'_j, \tilde{D}''_j, \tilde{D}'''_j \rangle, u, u'), \text{where} \\
\tilde{D}''_j = (D''_j)^{1/\tilde{P}(0)}, \tilde{D}'''_j = (D''_j)^{\tilde{P}(u')/\tilde{P}(0)}
\end{gathered}
$$

where $\tilde{P}$ is the polynomial in $\tilde{MK}$, and u' is Carol's identity.

- MA-Decrypt$(CT'', \tilde{SK})$. To decrypt a ciphertext CT using a delegated key, first Alice's proxy converts CT producing $\{C''_y = C'^{X_A}_y\}, \lambda_A$. Then Bob's proxy converts it further, based on identity u', to obtain $\{C'''_y = C'^{X_B}_y\}, \lambda_B$.

Bob's identity is conveyed to Carol as a part of the communication for $\tilde{SK}$, or some other way. Carol does a modified bilinear pairing in the *DecryptNode*, and finally decrypts the data.

$$
\begin{aligned}
&DecryptNode(CT, \tilde{SK}, x) \\
&= \frac{e(C_x, D_i)}{e(\tilde{D}''_i, C'''_i)^{\lambda_B} e(\tilde{D}'''_i, C'_x)^{\lambda_B \lambda_C} e(D'_i, C''_i)} \\
&= \frac{e(C_x, D_i)}{e(g_0, g_1)^{r_i h_x q_x(0) X_B \lambda_B P_A(B)/P_B(0)}} \\
&\cdot \frac{1}{e(g_0, g_1)^{r_i h_x q_x(0) P_A(B) P_B(C)/P_B(0) \lambda_B \lambda_C} e(D'_i, C''_{xA})} \\
&= \frac{e(g_0, g_1)^{r q_x(0) + r_i h_x q_x(0) P_A(0)}}{e(g_0, g_1)^{r_i h_x q_x(0) \lambda_B P_A(B)} e(g_0, g_1)^{r_i h_x q_x(0) X_A}} \\
&= \frac{e(g_0, g_1)^{r q_x(0) + r_i h_x q_x(0) P_A(0)}}{e(g_0, g_1)^{r_i h_x q_x(0) P_A(0)}} \\
&= e(g_0, g_1)^{r q_x(0)}
\end{aligned}
$$

The rest of the decryption proceeds as before. If Alice revokes Bob, or Bob revokes Carol, the decryption will not succeed since both Alice and Bob's proxies participate in the decryption, and the delegated secret key $\tilde{SK}$ contains information about all Alice, Bob, and Carol.

3.3.3 Attribute Revocation

In this section, we describe attribute revocation from a given secret key in a limited universe. This is useful since often the key authority may want to merely revoke a few attributes from her contacts instead of the whole key. For instance, in a social network, user A might want to remove colleague

attribute from B, but B still stays a friend.

Intuition:

The master secret key contains one polynomial P_i of degree t_i for each possible attribute i that the key authority defines in her universe. Any attribute can be introduced later by generating a new polynomial and storing it in MK. $P_i(0)$ is used to blind the corresponding attribute in the secret keys. Each user u also gets a random share $P_i(u)$ of $P_i(0)$ in her key. The proxy key consists of t_i such shares for each attribute in the policy used in the ciphertext. Whenever some attribute is revoked from some user, that share becomes a part of the proxy key, and hence the converted ciphertext. Therefore, the revoked user does not have enough points, i.e. $(t_i + 1)$ points for that specific attribute to unblind her key and the ciphertext, and thus fails to decrypt it. Non-revoked users always have $(t_i + 1)$ distinct shares for attribute i, and so decrypt successfully. When no attribute is revoked, the proxy key consists of t_i random points for each attribute i.

- Setup: The setup algorithm randomly generates a polynomial P_y of degree t_y over $\mathbb{Z}_p$ for each attribute $y \in Y$ where Y is the set of attributes in the system, sets the secret $P_y(0)$, and randomly chooses $\alpha, \beta \in \mathbb{Z}_p$. The public key PK and master secret key MK are generated as follows:

$$PK = (\mathbb{G}_1, \mathbb{G}_2, g_1, g_2, h = g_1^{\beta}, e(g_1, g_2)^{\alpha}), MK = (\beta, g_2^{\alpha}, \forall_{y \in Y} : P_y)$$

- KeyGen(MK, S, $\{P_x\}$): The algorithm KeyGen takes MK and a list of polynomials $\{P_x\}$ (possibly empty) for new attributes not in MK, and outputs the secret key corresponding to the set of attributes S for the user u_k. It also updates MK with the new attributes ($\{P_x\}$). The polynomial in each

key component is specific to the attribute represented by that component.

$$SK = (D, \forall j \in S : \langle D_j, D'_j, D''_j \rangle), \text{ where } D = g_2^{(\alpha+r)/\beta},$$
$$D_j = g_2^r \cdot H(j)^{r_j P_j(0)} = g_2^{r+h_j r_j P_j(0)}, D'_j = g_1^{r_j}, D''_j = (D'_j)^{P_j(u_k)} = g_1^{r_j P_j(u_k)}$$

- Encrypt(PK, M, τ): The encryption algorithm works as in key revocation.

- ProxyRekey(PK, MK, $\forall y \in Y : RL_y$): This algorithm takes a list of revoked users $RL_y = \{u_i\}, 1 \leq i \leq t_y$ for each attribute $y \in Y$, and generates the proxy key PXK. In case of fewer than t_y revocations, random values $\langle x, P_y(x) \rangle$ are generated to make RL_y of size t_y. The set of users from whom different attributes are revoked, may or may not overlap. Without loss of generality, we assume that the sets of revoked users don't overlap.

$$PXK = \forall y \in Y, \forall u_i \in RL_y : \langle u_i, P_y(u_i) \rangle$$

- Convert(PXK, $\forall y \in Y : C_y$): This algorithm is run by the proxy to convert the ciphertext components C'_y to C''_y for the user u_k. $\forall y \in Y$, the algorithm also calculates λ_k^y for u_k.

$$\lambda_i^y = \frac{u_k}{u_k - u_i} \cdot \prod_{j \neq i} \frac{u_j}{(u_j - u_i)}, \forall u_i, u_j \in RL_y, u_k \notin RL_y, RL_y \in PXK$$
$$\forall y \in Y : C''_y = (C'_y)^{\sum_{i=1}^{t_y} \lambda_i^y P_y(u_i)} = g_2^{h_y q_y(0) \sum_{i=1}^{t_y} \lambda_i^y P_y(u_i)}$$

- Decrypt(CT, SK): For each leaf node x where $i = attr(x)$, $i \in S$ (S is the set of attributes for which SK is issued), i is not revoked from u_k, and P_i is a polynomial of degree t_i for the attribute i, the algorithm proceeds as

follows:

$$
\begin{aligned}
DecryptNode(CT, SK, x) &= \frac{e(C_x, D_i)}{e(D_i'', C_x')^{\lambda_k^i} e(D_i', C_x'')} \\
&= \frac{e(g_1, g_2)^{rq_x(0)+h_i r_i P_i(0) q_x(0)}}{e(g_1, g_2)^{r_i h_i q_x(0)(\lambda_k^i P_i(u_k) + \sum_{j=1}^{t_i} \lambda_j^i P_i(u_j))}} \\
&= \frac{e(g_1, g_2)^{rq_x(0)+h_i r_i P_i(0) q_x(0)}}{e(g_1, g_2)^{r_i h_i q_x(0) P_i(0)}}, k \notin \{1, 2, \ldots, t_i\} \\
&= e(g_1, g_2)^{rq_x(0)}
\end{aligned}
$$

Otherwise *DecryptNode* returns $\perp$. The rest of the decryption proceeds as before. In summary, if an attribute i is revoked from user u, he can not do pairing on C_x'' and D_i'. He can continue to use components related to his other unrevoked attributes. Therefore, some of his attributes are revoked whereas some continue to be active.

3.3.4 Distribution of the Proxy

In our design, the proxy is semi-trusted. We assume that the proxy is updated with a new proxy key with each revocation, and it forgets the old keys. In order to enhance proxy security, we propose an alternate design to a centralized proxy. An alternate design to a centralized proxy is a *threshold-based distributed proxy.*

Threshold secret sharing can be used to split the proxy functionality among several parties. This model, where we can assume that majority of the parties are not actively malicious, will ensure the security of the proxy. With (t, n) threshold proxies, i.e., proxy functionality is distributed to n parties with t as the threshold value, $(t < n)$, $t + 1$ out of n proxies need to be malicious, i.e., colluding with the decryptor, to allow a revoked user to decrypt data.

3.4 Security Analysis

First, we need to define the requisite security properties for CP-ABE with Proxy Revocation. We present the definition for identity-based revocation; the definition for attribute-based revocation is analogous. We base our definition on the security model defined by Bethencourt et al. [49], with the addition of revocation and proxy operations. In this game, all encryptions remain secure even when the adversary compromises the proxy and obtains its key material, as long as this happens after the most recent revocation.

Setup. The challenger runs the SETUP algorithm and gives the public parameters, PK, to the adversary. The challenger also runs the PROXYREKEY$(PK, MK, \emptyset)$ algorithm to generate a proxy key PXK.

Phase 1. The adversary makes repeated queries to KEYGEN to obtain keys for users $u_1, \ldots, u_{q_1}$ with sets of attributes $S_1, \ldots, S_{q_1}$. The adversary also interacts with the proxy by calling CONVERT with the input $(\{C'_1, \ldots C'_r\}, u_k)$ for $C'_i \in \mathbb{G}_1$ and $u_k \in \mathbb{Z}_p$, at which point the challenger runs the CONVERT algorithm with the stored proxy key PXK. Finally, the adversary may call PROXYREKEY by supplying a revocation list RL. This will cause the challenger to update the proxy key PXK.

Challenge. The adversary submits two equal length messages M_0 and M_1 and an access structure $\mathbb{A}^*$. The adversary also supplies a new revocation list RL^*. RL^* and $\mathbb{A}^*$ satisfy the constraint that, for each user u_k, either $u_k \in RL^*$ or S_k does not satisfy $\mathbb{A}^*$.

The challenger flips a coin to obtain a random bit b and returns M_b encrypted with the access structure $\mathbb{A}^*$. Additionally, it runs PROXYREKEY (PK, MK, RL) and returns the resulting key PXK to the adversary.

Phase 2. The adversary makes repeated queries to KEYGEN to obtain keys

for users $u_{q_1+1}, \ldots, u_{q_2}$ with attribute sets $S_{q_1+1}, \ldots, S_{q_2}$. The new keys have to satisfy that if $u_k \notin RL^*$, then S_k does not satisfy $\mathbb{A}^*$.

Guess. The adversary outputs a guess b' of b.

The advantage of an adversary is defined as $Pr[b' = b] - \frac{1}{2}$.

Definition 1 *A ciphertext-policy attribute-based encryption with proxy revocation scheme is secure if all polynomial time adversaries have at most negligible advantage in the above game.*

3.4.1 Proof Sketch

We can prove the security of our scheme using a variant of a generic bilinear group model. Note that since the security of the original CP-ABE scheme relies on the generic bilinear group model, the assumption we make is only slightly stronger than the original. In particular, we must work within an asymmetric bilinear group, with a pairing of $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, such that there is no efficiently computable isomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$. (In a symmetric bilinear group, a user could submit D'_j to CONVERT and recover $g_0^{r_j P(0)}$, obviating the need to use the proxy in further decryptions.) This is believed to hold true for MNT curves [56].

The generic asymmetric bilinear group model. Consider three random encodings of the additive group $\mathbb{F}_p$ represented by injective maps $\psi_1, \psi_2, \psi_T : \mathbb{F}_p \to \{0,1\}^m$, where $m > 3 \log p$. We will define $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ as the range of the respective map. We are given access to a group action oracle for each group and an oracle for a non-degenerate bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ (we will refer to the ranges of ψ_1, ψ_2, ψ_T as $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, respectively). We are also given oracle access to the isomorphism $\phi : \mathbb{G}_2 \to \mathbb{G}_1$ and a hash function $H : \{0,1\}^* \to \mathbb{G}_2$. Finally, we let $g_i = \psi_i(1)$ for $i = 1, 2$.

Theorem 1 *The construction presented in Section 5.3 is secure under the generic asymmetric bilinear group model.*

We sketch the main argument here; most of the rest of the details are similar to the proof presented by Bethencourt et al. [49].

First of all, we can assume that no "unexpected collisions" happen between the maps, meaning that if we keep track of the algebraic expressions passed to ψ_1, ψ_2, ψ_T, and ϕ, two values are equal if and only if the expressions are symbolically equivalent. This assumption is true except for a negligible probability.

For simplicity of presentation, we will assume that $\mathbb{A}^*$ contains a single attribute A_j for some j. Then after phase 2, the adversary has the following elements available:

$\mathbb{G}_1 : g_1, g_1^{\beta}, C = g_1^{\beta s}, C_j = g_1^s (= g_1^{q_j(0)}), C_j' g_1^{h_j s},$

where s is the random secret used to encrypt the challenge message and h_j is implicitly defined such that $H(j) = g_2^{h_j}$.

$$\mathbb{G}_2 : g_2, D = g_2^{(\alpha + r_{u_k})/\beta}, D_j = g_2^{r_u + h_j r_{u_k,j} P(0)},$$
$$D_j = g_2^{r_{u_k,j}}, D_j'' = g_2^{r_{u_k,j} P(u_k)}$$

for each queried user u_k where $A_j \in S_k$. Note that we can ignore all $D_{j'}$ for $j' \neq j$ because, as with CP-ABE, they will not help with decryption.

$$\mathbb{G}_T : e(g_1, g_2)^{\alpha}, M \cdot e(g_1, g_2)^{\alpha} s$$

In addition, the adversary knows $u_k, P(u_k)$ for all the revoked users in RL^*. Note that we can ignore any other elements of $\mathbb{G}_1$ obtained through calls to CONVERT during Phase 1, or through calls to the isomorphism. This

is because in order to guess correctly with a non-negligible probability, the adversary needs to compute $e(g_1, g_2)^{\alpha s}$. Since there are no occurrences of s in $\mathbb{G}_2$, each pairing must involve g_1^{sk} for some k, and hence be derived from C, C_j, or C'_j.

The adversary can compute $e(C, D^{(u)}) = e(g_1, g_2)^{\alpha s + r_u s}$, hence computing $e(g_1, g_2)^{\alpha s}$ is equivalent to computing $e(g_1, g_2)^{r_u s}$ for some user u. Note also that the secret keys obtained for other users are *not* helpful here, for the same reason as in original CP-ABE. Algebraically, the adversary must solve the following equation:

$$e(g_1, g_2)^{r_u s} = e\left(g_1^x, \left(D_j^{(u)}\right)^y\right) e(g_1, g_2)^z$$

where x, y, and z are derived from the available elements *other* than $D_j^{(u)}$. Note that $x, y \neq 0$, since otherwise there is no way to introduce r_u into the right-hand side. However, the rest of the values are *independent* of $P(0)$, since the only values of the polynomial available to the adversary outside of $D_j^{(u)}$ are $P(u_k)$ for $u_k \in RL^*$, and thus are not sufficient to determine $P(0)$.

3.5 Implementation and Evaluation

We implemented the constructions in EASiER, as described in Section 5.3. We took the BSW CP-ABE toolkit [57] as base, and implemented the revocation scheme. The current implementation supports *key revocation* and *access delegation* for the multiple authorities setting. Similar techniques can be applied to perform attribute revocation.

The implementation uses MNT curves [58] with 159-bit base field. The experiments were performed on an emulab [1] d710 machine with 2.40 GHz

[1] http://www.emulab.net

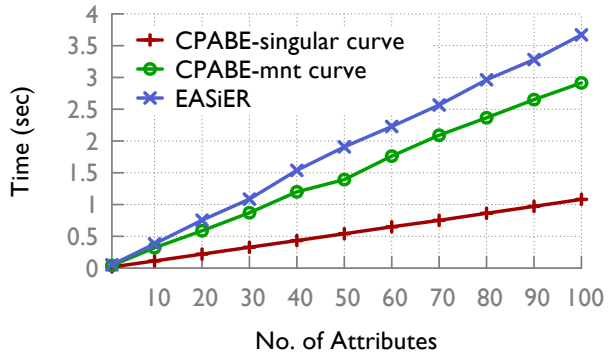
64-bit Quad Core Xeon processor running Ubuntu 10.04. We also implemented a Facebook application to provide the functionality on a social network. The code for EASiER and the Facebook application are available at `https://bitbucket.org/hatswitch/easier` and `https://bitbucket.org/hatswitch/easier-fb` respectively.

3.5.1 Performance Analysis

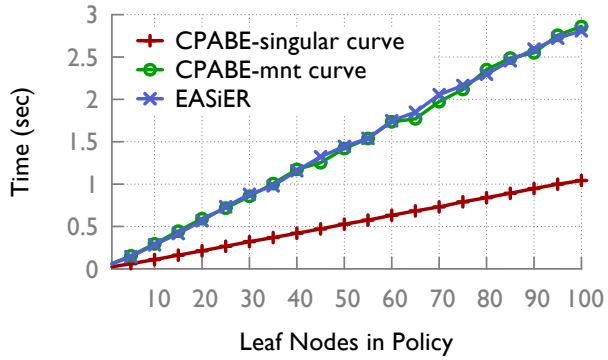
We provide some information on the performance evaluation of EASiER, and compare it with BSW CP-ABE. Though CP-ABE implementation uses symmetric pairing, we use asymmetric pairing for both EASiER and CP-ABE in our implementation. This provides security by preventing key and ciphertext components exchange. The comparison results are shown in Figure 3.2, revocation/proxy re-keying time and the time to convert ciphertext by the proxy in EASiER are shown in Figure 3.3, and finally, the time to delegate keys and decrypt with a delegated key are shown in Figure 3.4.

Key Generation: Key generation time is linear with number of attributes both in BSW CP-ABE and EASiER. EASiER does extra exponentiation, and generates an extra component for each attribute. BSW CP-ABE with supersingular curve requires the least time as expected.

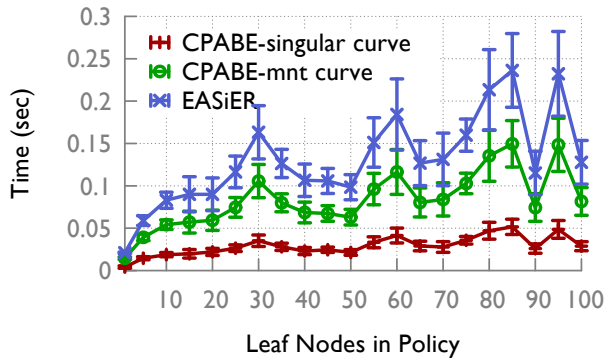
Encryption: To test encryption and decryption, we generated 10 different policies with a random combination of *and*s and *or*s, and attributes randomly chosen from $\{attr_1, attr_2, \ldots, attr_{100}\}$ for each of the desired number of leaves $(1, 5, 10, \ldots 100)$. An example policy is "$attr_1$ and ($attr_2$ or $attr_3$ or ($attr_4$ and $attr_5$))". Encryption is also linear with respect to the number of leaf nodes in the policy. Since encryption is same in both the schemes, BSW CP-ABE with MNT and EASiER have the same performance.



(a) Key Generation



(b) Encryption



(c) Decryption

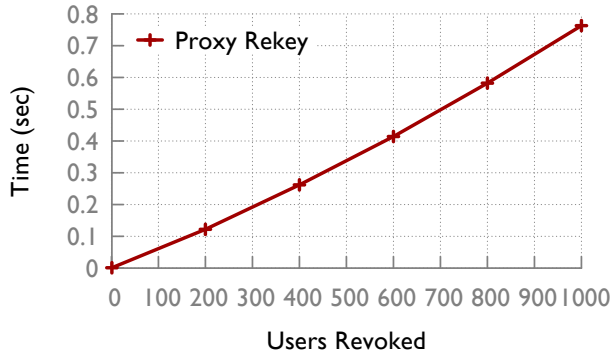
Figure 3.2: Performance Analysis: Comparison of EASiER with BSW-CPABE. Number of the revoked users is set to 500. (a) Secret key generation depends on the number of attributes in the key. (b) Encryption depends on the number of leaf nodes in the policy. (c) Decryption depends on the type of policy used to encrypt and the number of leaf nodes in the policy.

Decryption: Decryption depends on the policy used in encryption, and the attributes involved. We generated a decryption key with 100 attributes, i.e., the superset of all the attributes used to generate the policies, and so it satisfies each of the policies used in encryption. The decryption results are shown with a 95% confidence interval. All the lines show the performance when an optimization was used to ensure the usage of the minimum number of leaves in the algorithm *DecryptNode*.

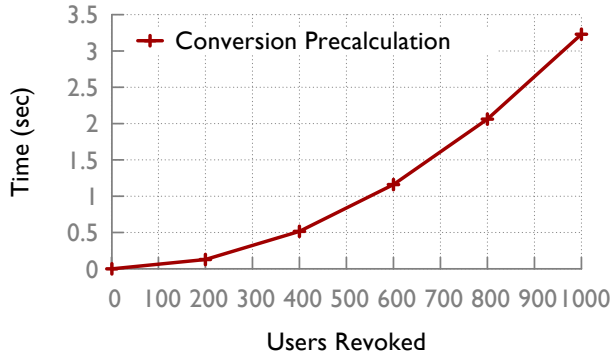
Proxy Rekey and Conversion: EASiER involves two extra costs before decryption: re-keying the proxy and converting the ciphertext components specific to the leaves in the policy. The re-keying results (Figure 3.3a) show that for even 1 000 revoked users, the time required is around 0.8 second. This should be compared with the time required to rekey the rest of the group, i.e., generate a new key for everyone, when even one person in the group is revoked. The latter number depends on group size and number of attributes in the key. For example, to rekey a group of size 100 each with a key size of 20 attributes in CP-ABE-singular curve setting (Figure 3.2a), the time will be 22 seconds (100×0.22).

Conversion primarily involves one exponentiation for each of the leaf specific ciphertext components. It also calculates λ_k for the requester u_k, and completes the $\lambda_i s$ for each of the revoked users. We perform an optimization by allowing the proxy to pre-calculate a portion of the λ_i's in the CONVERT algorithm. With the optimization, the proxy needs to do 1 multiplication per revoked user to calculate λ_i. Let RL be the list of revoked user. It works as follows:

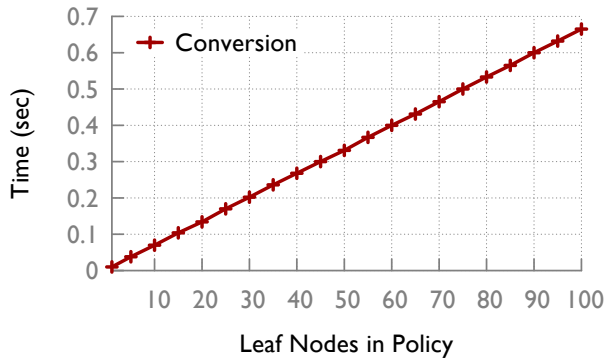
$$\lambda_i' = \prod_{u_i, u_j \in RL, i \neq j} \frac{u_j}{(u_j - u_i)}, \text{ and } l_i' = \lambda_i' P(u_i)$$



(a) Revocation/Proxy Rekeying



(b) Conversion Pre-calculation by Proxy



(c) Ciphertext Conversion by Proxy

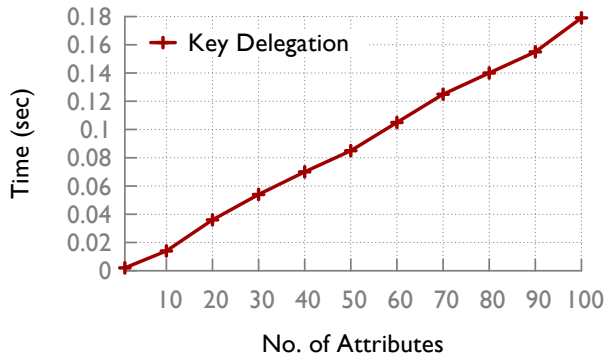
Figure 3.3: Performance Analysis: Time Required for Revocation/Rekeying the Proxy and Ciphertext Conversion. (a) Generating proxy key depends on the number of revoked users. (b) Partial pre-calculation of ciphertext conversion depends on the number of users revoked (c) Ciphertext conversion depends on the number of leaf nodes in policy and is hardly affected by the number of users revoked.

$$l_i = l'_i \frac{u_k}{(u_k - u_i)} = \lambda'_i \frac{u_k}{(u_k - u_i)} P(u_i) = \lambda_i P(u_i), \forall\ u_i \in RL, u_k \notin RL$$

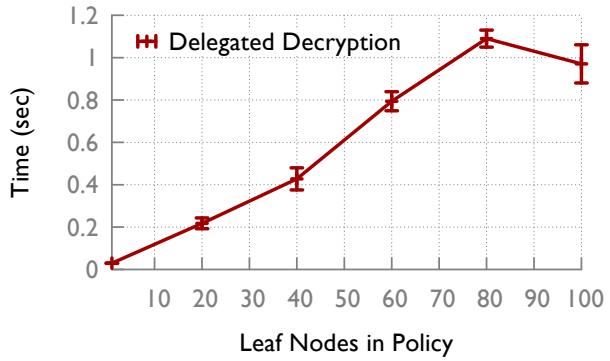
Figure 3.3b shows the time required for the proxy to do the pre-calculation. Note that this is a one-time cost each time the proxy is re-keyed and is not faced by users. Figure 3.3c shows the time required to actually convert a ciphertext. The result depends on the number of leaf nodes and is hardly affected by the number of revoked users, since the time to do exponentiations for the leaf nodes dominates the time to do t multiplications to calculate λ_is. Figure 3.3c shows the time for 1000 revoked users. Conversion of each leaf node can be performed in parallel, so using a multi-core machine or a cloud-based service can drastically reduce the delay experienced by users during this operation. A user performing decryption only faces the conversion time shown in Figure 3.3c along with the decryption time mentioned earlier.

Key Delegation: We generated a key with 100 attributes and delegated various number of attributes from this key. The results are shown in Figure 3.4a. The delegation time depends on the number of attributes delegated. To delegate 100 attributes from this key, it requires 0.18 second.

Decryption with Delegated Key: As in regular decryption, decryption with a delegated key depends on the policy used in encryption, and the attributes involved (Figure 3.4b). We used the same ciphertexts that we used in regular decryption. We decrypted the ciphertexts with the delegated keys that we generated by delegating various number of attributes from a key with 100 attributes. The results are shown with a 95% confidence interval. Note that, to decrypt with a delegated key, a user has to face two additional time costs for ciphertext conversion by two different proxies.



(a) Key Delegation



(b) Decryption with Delegated Key

Figure 3.4: Performance Analysis: Time Required for Key Delegation and Decryption with Delegated Key. The number of revoked users is set to 500. (a) Key delegation depends on the number of delegated attributes. (b) Decryption with delegated key depends on the type of policy used to encrypt and the number of leaf nodes in the policy.

Component Size and Communication Overhead: Table 3.1 shows the sizes of the components involved in the system. Elements from $\mathbb{G}_0, \mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{Z}_p$ require 44, 124, 124, and 24 bytes respectively to represent (Table 3.1a). Users have to communicate with the proxy for conversion by sending C'_y, and receiving C''_y. These are represented using elements from $\mathbb{G}_1$. This requires 124 bytes to represent (120 for the actual data, and 4 for the variable size). Hence, conversion of a ciphertext with l leaf nodes in the policy will need to transfer $124l$ bytes each way. The user also sends u_k, and receives λ_k back. These are represented using $\mathbb{Z}_p$ which requires 24 bytes.

Public Key consists of a string describing the pairing used (980 bytes), g_0 and h from $\mathbb{G}_0$, g_1 from $\mathbb{G}_1$, and $e(g_0, g_1)^\alpha$ from $\mathbb{G}_2$. Master Key consists of β from $\mathbb{Z}_p$, and ${g_1}^\alpha$ from $\mathbb{G}_1$ in CP-ABE. In EASiER, it also consists of a polynomial of degree t. The polynomial consists of an integer t, and $t + 1$ coefficients from $\mathbb{Z}_p$. Private Key consists of D from $\mathbb{G}_1$, number of attributes n (integer), and n of $\langle D_j, D'_j \rangle$s from $\mathbb{G}_1$ and $\mathbb{G}_0$ respectively and attributes of length a (string) . EASiER also contains n of D''_j from $\mathbb{G}_0$. Ciphertext consists of $\tilde{C}$ from $\mathbb{G}_2$, C from $\mathbb{G}_0$, and components for each node in the policy. Both intermediate (i nodes) and leaf (l nodes) nodes have a threshold k (integer), and number of children (0 for leaf, also an integer). A leaf node has a string attribute of length a, and C_y and C'_y from $\mathbb{G}_0$ and $\mathbb{G}_1$ respectively. Proxy Key consists of t $\mathbb{Z}_p$ elements.

Attribute Revocation: We can estimate the time required to perform attribute revocation. All the algorithms work similarly. The only extra work is in `Setup` where instead of just one polynomial, a polynomial is generated for each attribute in the system. It requires 0.34sec to generate a polynomial of degree 500 and increases linearly with the degree of the polynomial. Hence, generating a polynomial for each attribute is scalable.

Table 3.1: Component Size and Communication Overhead in EASiER

(a) Element Size

Group	**Size (bytes)**
$\mathbb{G}_1$	44
$\mathbb{G}_2$	124
$\mathbb{G}_T$	124
$\mathbb{Z}_p$	24

(b) Component Size

Component	**EASiER (bytes)**	**CP-ABE (bytes)**
Public Key	1316	1316
Master Key	$152 + (t+1)24$	148
Private Key	$128 + (a+212)n$	$128 + (a+168)n$
Ciphertext	$168 + 8i + (176+a)l$	$168 + 8i + (176+a)l$
Proxy Key	$24t$	NA
C''_y	$124l$	NA

3.5.2 Facebook Application

Finally, we developed a Facebook application for our protocol. In the current implementation, all protocol functions are performed at our application server. Moreover for convenience, we chose the client's proxy server to be the application server itself. When a user first installs our Facebook application, an account is created in the application database. The application retrieves public profile information of the users who install it using Facebook API, and uses this data as its user profile information. It is available at `http://apps.facebook.com/myeasier/`.

3.6 Related Work

3.6.1 Revocation Schemes

Attrapadung and Imai address the revocation problem in [59] by combining broadcast encryption scheme with both CP-ABE and KP-ABE. This scheme requires knowledge about the list of all possible users during encryption, i.e., attaches one component per user to the ciphertext. Knowing the list of all possible users in advance is inconvenient for most scenarios, specifically OSNs. Bethencourt et al. in [49] and Boldyreva et al. in [60] propose expiration-based revocation scheme for CP-ABE and KP-ABE respectively. In these schemes, the secret keys (CP-ABE) or the ciphertext (KP-ABE) contain expiry time as an extra attribute. Time-based revocation may not be a desired property in several applications where an immediate revocation is necessary. It introduces a window of vulnerability, i.e., the gap between the desired time from revocation to the actual time of revocation. Ostrovsky et al. [61] present a new KP-ABE scheme with non-monotonic access structure to support negation of attributes. Revocation can be implemented by adding a NOT-attribute to the policy in private key. However, this will require re-keying the users from whom an attribute is revoked.

Lewko et al. in [62] propose a revocation scheme with small private keys. However, their approach also increases the size of the ciphertext by incorporating the list of revoked and unrevoked users with it. In [63] Hur proposes a revocation scheme for CP-ABE where each leaf node, i.e. attribute in the policy used to encrypt the data contains a group of users who possess that attribute. The scheme consists of a key generation center (KGC) and a data storing center (DSC). The DSC is equivalent to the proxy in our proposed scheme. However, the approach requires secret key update for all the users

of an attribute group from which at least one user was revoked; this also includes the non-revoked users in that group.

The CCA secure construction of Yu et al. [64] involves updating the master key component of each attribute that has been revoked in the system. The public key components are then updated and data is encrypted with the new public key. Finally, proxy re-keys are generated that enable a proxy to update the user secret keys to the new version for all but the revoked user. Basically the re-keying burden is placed on the proxy, instead of users. We note that the existing data would need to be re-encrypted by the proxy; placing a significant burden on the system. While our scheme is only CPA secure under the generic group model (weaker security than Yu et al.), it does not require re-encryption of existing data.

3.6.2 Proxy-based Re-encryption

Our revocation techniques are based on the notion of proxy re-encryption. We will now briefly trace some developments in this field, and discuss why the state of the art techniques cannot be directly applied to our setting.

Blaze et al. [65] introduced the notion of *proxy re-encryption*, in which a proxy could convert a ciphertext for Alice into a ciphertext for Bob, using a specially generated proxy key. The holders of public-key pairs A (Alice) and B (Bob) publish a proxy key $\pi_{A \to B}$ such that $D(\pi(E(m, e_A), \pi_{A \to B}), d_B) = m$, where $E(m, e)$ is the public encryption function of message m under encryption key e, $D(c, d)$ is the decryption function of ciphertext c under decryption key d, $\pi(c, \pi_{A \to B})$ is the proxy function that converts ciphertext c according to proxy key $\pi_{A \to B}$, and e_A, e_B, d_A, d_B are the public encryption and secret decryption component keys for key pairs A and B, respectively. In

the El-Gamal based construction proposed by Blaze et al., while the proxy cannot see the plaintext message m, it can collude with B to recover the secret key for A. Moreover, the construction is *bidirectional*, and the proxy key can be used to convert ciphertext for Bob into a ciphertext for Alice as well.

Ateniese et al. [66] propose *unidirectional* protocols for proxy re-encryption based on bilinear maps, where a re-encryption key from A to B does not imply a re-encryption key from B to A. Canetti and Hohenberger [67] proposed the first CCA secure bidirectional proxy re-encryption scheme, while Libert and Vergnaud [68] proposed the first CCA secure unidirectional proxy re-encryption scheme in the standard model. We note that all of the above schemes are limited to the public key setting. Green and Ateniese [69] extended the model to the identity based setting by proposing a scheme for identity based proxy re-encryption, but it was not until Liang et al. [70] that the attribute based encryption setting was considered.

In the attribute based setting of Liang et al., a user could designate a proxy, who can re-encrypt a ciphertext with a certain access policy into another ciphertext with a different access policy. Furthermore, the authors showed their scheme to be selective-structure chosen plaintext secure. However, it is not possible to apply their construction to the problem of attribute revocation because their construction does not support negative attributes.

3.7 Conclusion

We present an access control architecture for Online Social Networks, named EASiER, that supports efficient revocation in attribute-based encryption. We achieve this revocation scheme by introducing a semi-trusted proxy, lever-

aging ideas from a group communication scheme, and combining it with ABE. Although we showed our approach in an OSN setting, it can be applied to any context where ABE is used for data protection with dynamic group membership. We implemented the scheme and compared it with Bethencourt et al.'s Ciphertext-Policy Attribute-Based Encryption (CP-ABE). Our results show that EASiER is scalable in terms of computation and communication for OSNs; accordingly, we have built a prototype and a Facebook application to provide such encryption.

4 DECENT

Our second proposed approach is called DECENT, an architecture for enforcing access control in a decentralized OSN. Our focus is on providing data confidentiality, integrity, and availability in the presence of malicious nodes in a distributed setting. Our architecture is also able to protect the privacy of user relationships. Users consider their social contacts to be sensitive information, evidenced by the public outcry when Google Buzz made this information public, and was forced to change its default settings [29]. Alternative decentralized designs [23–26] remove reliance on a central entity, but have not adequately tackled confidentiality and access control issues.

DECENT is based around a flexible object-oriented design (OOD) that supports the main functionality of OSNs and captures the complex multi-principal interactions that are common in social networks. The confidentiality and integrity of data are protected by a cryptographic mechanism so that they can be stored in untrusted nodes in a distributed hash table (DHT). The standard DHT mechanisms are extended to ensure availability despite malicious attempts to erase or overwrite stored data.

Our contribution is twofold:

1) Design: We propose a decentralized OSN architecture that: i) provides flexibility in data management through OOD; ii) uses an appropriate and advanced cryptographic scheme that supports efficient access revocation and fine-grained policies on each piece of data; and iii) combines confidentiality, integrity, and availability by using the functionalities of a DHT—all

the existing designs focus on one or two, but not all of these aspects. The novelty of our architecture lies in integration of existing primitives that are tailored to enhance the security and privacy of OSNs.

2) Prototype: We develop a prototype of DECENT—the wall and news-feed functionalities, to be specific—and evaluate its performance through simulation and experiments on PlanetLab. We evaluate DECENT using a FreePastry simulator and a Kademlia implementation on PlanetLab [71, 72].

Our analysis shows that the performance overhead of DECENT is acceptable, and that our architecture for privacy-preserving decentralized OSNs is feasible. For example, although a wall with 60 objects needs 90 s to be viewed by its owner, contents can be displayed progressively; i.e., older messages can be fetched while the user is reading the most recent messages, which are loaded within the first few seconds. Our architecture thus demonstrates that existing security primitives with well understood properties can be leveraged to provide a compelling privacy-preserving alternative to centralized OSNs.

4.1 Functional Model

Much of the functionality of OSNs can be described as users posting content and their social contacts viewing, commenting on, and annotating such content. To provide a flexible, general model of these operations, we define a *container object* that has two components: the main content and a list of comments/annotations, represented as references to other container objects. The main content can take on many types, such as a status update, a shared link, a photo or video, or a collection of container objects (e.g., a photo album). For our purpose, the content type is not important; the key difference is that the *access permissions* on the comments can be more restrictive than

the content to enforce the policy of each object and owner individually.

A user's profile is a root object, which contains references to other objects, such as contact information, a wall, photo albums, etc. Similarly, other objects may consist of some content and references to other objects. For example, a wall may have references to status messages and posts where a status/post object may contain references to comment objects in addition to the status data. Thus, each user's content is organized in a hierarchical fashion (although we do not enforce a tree structure—a single object may be referenced by multiple "parent" objects). With this degree of granularity in our object design, access permissions can be assigned specifically for each object and then referenced by other containers.

4.2 Architecture

DECENT is a decentralized OSN, which employs a DHT to store and retrieve data objects created by their owners. Each object is encrypted to provide confidentiality. The primary advantage of our architecture is its modularity, i.e., the data objects, the cryptographic mechanisms, and the DHT are three separate components, interacting with each other through well-defined interfaces. The modular design allows us to separate the various cryptographic implementations (ABE, signatures) from the basic object model (container objects, permissions, etc.) and use any type of DHT.

4.2.1 Access Policies

Each object has three access policies associated with it. The policies are either attribute-based (AB), identity-based (IB), or a combination of both types. AB policies take the standard form of policies described through var-

ious attributes, for example, *friend*, *family*, *coworker*. AB policies can represent formulas over attributes, using operators such as $\wedge$, $\vee$, and k-of-n. Examples of AB policy are: $(friend \wedge coworker) \vee family$, and 2 *of* $\{friend, family, coworker\}$ (background on Attribute-based Encryption (ABE) is provided in section 4.2.2).

- Read policy (*R-Policy*) describes who may read the contents of the object. It is an AB policy that describes the attribute combination required for a user to decrypt an object's data.
- Write policy (*W-Policy*) describes who may modify the contents of the object or delete the object. It is an IB policy, which generally is set to the owner of the object.
- Append policy (*A-Policy*) describes who may add a comment/annotation to the object. It is also an AB policy.

These policies are defined by the owner at the time of object creation and are stored in the object metadata. The read policy is enforced through the use of cryptography. The write and append policies are enforced through a combination of cryptography and specialized DHT functionality (mentioned later). A reader can cryptographically verify the integrity of the object and be sure that the content and comments have been posted by parties authorized by the write and append policies, respectively. Additionally, the DHT storage nodes require authorization before each write operation to prevent malicious deletions and vandalism.

The authorization does not reveal a user's identity, hence the storage node is not aware of the identities of users storing or retrieving data from it, and therefore a user's social graph is hidden from the storage nodes. DHT nodes

also implement a special append operation that adds a new annotation to the object while leaving existing content unmodified. When objects are being stored at malicious nodes, confidentiality is still preserved due to cryptography. However, the malicious nodes can impact the integrity and availability guarantees by deviating from the protocol, e.g., by deleting objects and/or returning previous versions of objects. Malicious nodes in the DHT can be tolerated using replication.

We support *fine-grained policies* meaning that a separate policy can be enforced on each object, access revocation is efficient, and comments have more restrictive policies than the parent object. For example, when Bob comments on Alice's status, the comment may have its own policy different than Alice's policy on the status object. To view the comment, a viewer has to satisfy both Alice and Bob's policy. A chain of comments can be created through this approach. This feature allows to control access to a comment even when it is posted on someone else's object.

Policies may change over time. If an append policy changes, a mechanism will verify the integrity of previously legitimate comments that satisfy the old policy. After the policy change, the object owner signs the old comments to ensure her readers that the old comments are valid. Changing the write policy is straightforward and needs simple update in the reference. For a read policy change that revokes access, no change in the object is necessary. However, if a read policy changes completely, the owner has to re-encrypt the data.

4.2.2 Cryptographic Protection

Objects are stored on untrusted DHT nodes, thus necessitating the use of cryptography to protect their confidentiality and integrity. For confidentiality, Baden et al. [73] observed that ABE [49] is a good fit for OSNs because it allows users to specify access policies in terms of groups of contacts, such as friends, family, coworkers, etc. We adopt several modifications to the base ABE to better satisfy our security requirements.

Confidentiality: We use an updated version of EASiER as the underlying ABE scheme. Like most public-key schemes, ABE is usually used in hybrid encryption mode, wherein the message is encrypted with a randomly chosen symmetric encryption key, which is in turn encrypted with ABE. We follow this approach with a modification that the ABEncrypted symmetric key is in fact part of the object reference and not included in the object itself. The main motivation for this choice is that the version of ABE we are using lacks policy privacy and this approach keeps the policy hidden from untrusted storage nodes. The reference is a part of the parent object and is encrypted with its symmetric key. As a result, the reference cannot reveal the policy.

An additional consequence of this approach is that when several references for an object exist, the object may have different read policies associated with it. For example, if Bob posts a comment on a status update on Alice's wall, he may wish to add a reference to his comment (or even the status update) to his own wall so that it can be seen by his contacts.[1] In this case, Bob's comment may be visible to some subset of Alice's contacts when reached through her status update, and some subset of Bob's contacts when reached

[1]Note that this may contravene Alice's privacy wishes, but no architectural protection short of DRM can prevent Bob from such re-sharing. A possible mitigation strategy is to issue a warning to Bob about the possible privacy breach, as is currently implemented in Google+.

through his wall.

Integrity: To protect the integrity of the object, we make use of digital signatures. The owner of the object digitally signs its contents and metadata, excluding the list of comments; each comment reference is signed individually using a different signature key. The write and append policies are implemented by controlling access to the corresponding secret signature keys, which act as *capabilities*, similar in spirit to Tahoe-LAFS [74]. The append-policy secret key is encrypted with an ABE policy; the write-policy key can be encrypted with conventional encryption since it typically needs to be accessible by a single person (the object owner). Both public keys are stored as part of the object metadata, but the write-policy public key must be replicated as part of the object reference to ensure its authenticity; the append-policy public key is authenticated by the write-policy signature. An object reference, therefore, consists of:

$$objRef \stackrel{\text{def}}{=} (objID, \text{ABE}(K, P), WPK)$$

where *objID* is a random object identifier, used to locate it in the DHT, K is the symmetric key used to encrypt the referenced object, P is the attribute-based read policy, and WPK is the write-policy signature public key. As an optimization, $\text{ABE}(K, P)$, the ABEncrypted symmetric key, can be replaced by an unencrypted K if the read policy of the referenced object matches the container.

4.2.3 Distributed Hash Table

Participants in the OSN are organized into a distributed hash table (DHT), such as Pastry [71] or Kademlia [72]. The DHT creates a scalable key-value store with an efficient lookup mechanism to locate nodes that store a given object. DHT lookups can be made secure against attacks [30], ensuring that a lookup will find the correct copy of an object if it exists. (It may additionally find incorrect copies provided by malicious nodes; cryptographic integrity protection described above can be used to identify the correct one.)

Objects in DECENT are stored in the DHT using the *objID* as the key. Each object ID is randomly chosen,[2] thus the user who runs the node storing an object has no relationship with the object owner and is therefore untrusted, but the object confidentiality and integrity is protected by cryptographic mechanisms outlined above. To ensure availability despite node churn and malicious attacks, several replicas of an object are maintained. DHTs typically use the neighbor set of the node responsible for the object key to maintain replicas; the number of replicas needs to be tuned based on the churn patterns of the network (malicious nodes can also be modeled as churn in this case), which we will study in our future evaluation. To guarantee freshness, each object has a version number as part of its metadata. The version number is authenticated by the write-policy signature, thus a user can query all of the replicas and use the freshest object returned.

Malicious users may try to modify or delete an existing object. Note that the write policy prevents them from creating modifications that will be accepted by the readers, as they cannot produce a correct signature, but they may overwrite and destroy legitimate data. To address this, the write-policy

[2]DHTs typically use 128- or 160-bit keys; this ensures that collisions between two randomly chosen keys are essentially impossible.

public key is stored unencrypted as part of the object metadata. The storage node will refuse to overwrite the stored object unless the new data is properly signed by the write-policy key; deletions must likewise be authenticated with a signature. Thus, as long as there is always at least one honest (but curious) replica for an object, it will persist despite any malicious attacks. The write-policy public key *should not* be a user's permanent public key, as otherwise a storage node could use it to link an object to its owner. Instead, a separate WPK is generated for each object, ensuring unlinkability of objects. A copy of the corresponding secret key (WSK) is stored in the object, encrypted with the owner's secret key. We use the Digital Signature Algorithm (DSA) for write-policy signatures because it allows new keys to be generated very quickly.

In addition to the standard *get* (read) and *put* (write) requests, the DECENT DHT supports an *append* request, which is used to add a comment reference to an existing object. As with write requests, the storage node verifies that the append carries the correct signature from the append-policy key, using the public key (APK) that is stored in the clear as part of the object.

4.2.4 Example

Join: To join DECENT, Alice sets up her profile, wall, and keys. She generates her ABE public and master secret keys and signature key pair. Alice creates an object with her profile information, encrypts it with a symmetric key, and signs it with her write-policy signature key SPK_{Alice}. She generates a random ID, saves her profile in the DHT using this ID, ABEncrypts the symmetric key with profile R-policy, and creates a reference to the profile

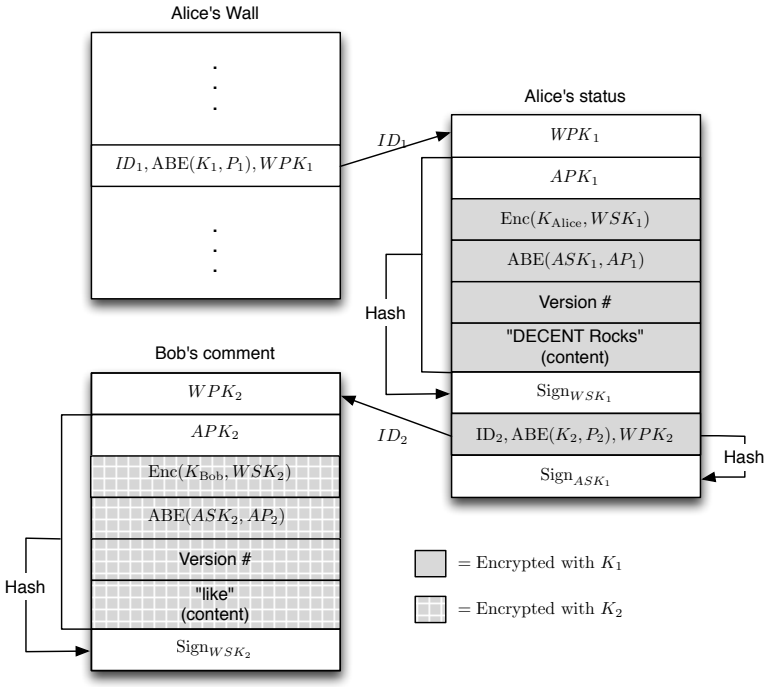


Figure 4.1: Example objects in DECENT. Alice's wall is represented as an object. Alice creates a status, stores it in the DHT and posts a reference to the status on her wall. To comment on this status, Bob creates a comment object, stores it in the DHT, and posts a signed reference to this comment on Alice's status object.

object. Similarly, she sets up her wall. Alice can be reached through the reference to a root object, which contains the references to her profile and wall. The root object acts as another regular object and is stored in the DHT. The root object can be thought of a user's landing page on Facebook.

Establish Contacts: To establish the relationship *friend, co-worker* with Bob, Alice generates an ABE secret key for Bob with the attributes *friend, co-worker*. Relationships are asymmetric, so Bob may establish just *acquaintance* relationship with Alice by issuing an ABE secret key for this attribute. Keys are exchanged out of band. Alice and Bob also exchange their root object references.

Post and Comment: Figure 4.1 shows an example object structure. When Alice wants to post a status update to her wall, she creates the status update object, complete with version number, contents, and public and secret keys for the write and append policies (WPK_1, WSK_1, APK_1, ASK_1). She then generates a signature over the write-policy signature key (WSK_1). She then picks a random symmetric encryption key K_1 and encrypts the object (except for WPK_1 and APK_1 and the signature); she also chooses a random id ID_1 and uses this to insert the object into the DHT. Finally, she creates a reference to the status update, including ID_1, K_1 and her write-policy public key (WPK_1) and adds it to her wall. K_1 is encrypted with an attribute-based policy P_1, which governs who can read the status update. Note that Alice's wall will also be encrypted with a symmetric key K_0, which is part of the reference to Alice's profile that she gives to her friends.

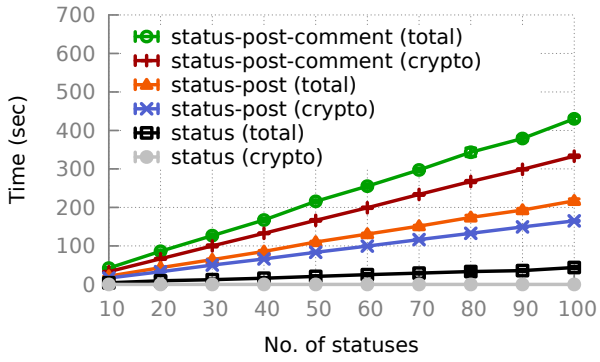
When Bob wants to read Alice's update, he finds the reference on Alice's wall and decrypts K_1 with his attribute-based secret key that he got from Alice, assuming that his attributes satisfy the policy P_1. Note that Bob has to satisfy the R-Policy associated with the wall object itself to get access to

the reference. He then retrieves the object from the DHT with the key ID_1 and decrypts the encrypted fields using K_1. Finally, he verifies the signature to ensure the authenticity of the object.

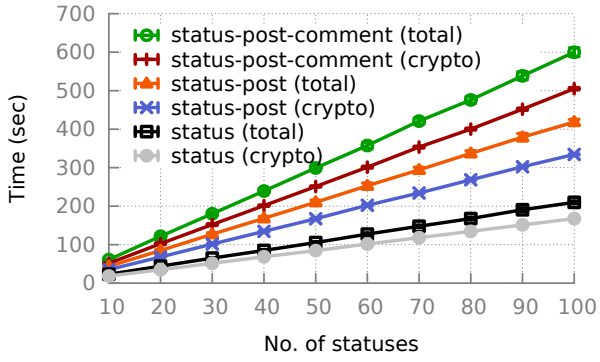
If Bob further wants to comment, then he first creates a comment object following a process similar to Alice's creation of her update. He then uses the *append* operation to insert a reference to the new object into Alice's update. Assuming he satisfies the A-policy AP_1, Bob decrypts ASK_1 and uses it to generate a signature on the reference. The encryption key K_2 is further ABEncrypted using Bob's policy P_2; thus, only users who satisfy both P_1 and P_2 will be able to read the comment.

4.3 Implementation and Evaluation

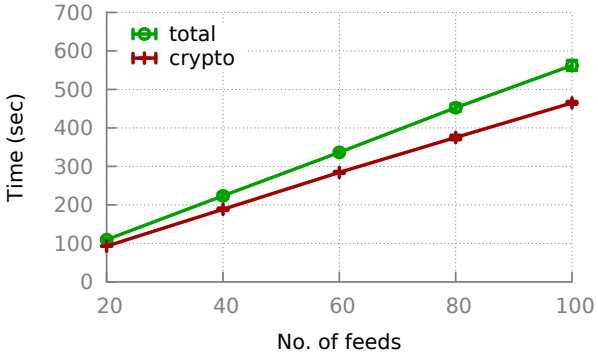
We have implemented a preliminary prototype of DECENT, which provides functionality similar to the Facebook wall. It also provides a basic newsfeed option, summarizing status updates from a person's contact/friend list. We use four different types of cryptographic schemes in DECENT: EASiER for ABE, AES for symmetric encryption, DSA for signatures, and RSA to encrypt the write policy signature key. We use a combination of EASiER and DSA to realize ABS. The key sizes are chosen as recommended by NIST [75] for maximum security (refer to table 4.1). We use FreePastry with Euclidean network topology for simulation, and Kademlia [47, 76] for the experiments on PlanetLab as the underlying DHT. Our proxy was run on a standard server for simulation and PlanetLab experiments.



(a) View My Wall



(b) View Others' Wall

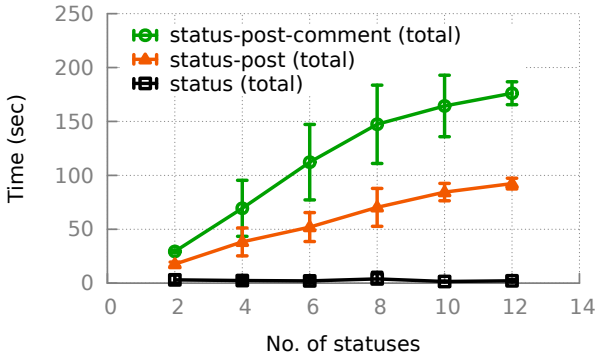


(c) View Newsfeed

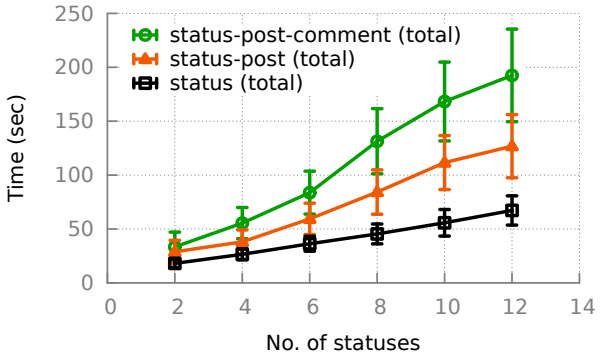
Figure 4.2: Simulation Results for 10,000 nodes: The average time to view another user's wall with 10 and 20 status/post/comments is about 60 s and 120 s respectively. The average time to view a newsfeed with 20 peers is 109 s.

Table 4.1: Key Size for Cryptographic Schemes used in DECENT

Cryptographic Scheme	**Type**	**Key Size (bits)**
Attribute-based Encryption	EASiER	MNT with 159
Symmetric Key Encryption	AES	128
Asymmetric Key Encryption	RSA	2048
Signatures	DSA	2048



(a) View My Wall



(b) View Others' Wall

Figure 4.3: PlanetLab Results: The average time to view another user's wall with 10 status/post/comments is about 168 s.

4.3.1 Simulation

We perform experiments to measure the performance of viewing a newsfeed and wall with varying numbers of status messages, posts, and comments. The simulation was run on a peer-to-peer network of 10 000 nodes. Figure 4.2 shows our simulation results.

View Wall: To view a wall, a user uses the wall reference to fetch the wall object, ABDecrypts the wall reference to get the AES key to decrypt the wall object, and gets the wall metadata and the references to statuses and posts. Each reference is used to fetch the corresponding status or post. The user ABDecrypts the reference to view the content of the object.

We perform tests to view wall with: 1) only statuses, 2) statuses and same number of posts from friends, and 3) statuses, posts, and one comment on each status from friends. Figures 4.2a and 4.2b show results for viewing one's own wall and friends' walls respectively. A data point such as x statuses means that when a user views a wall that contains statuses, posts, and comments, she views x of each, i.e., $3x$ objects.

We allow users to cache the AES encryption keys for the objects they create and thus avoid ABDecryption of the references to their own objects. Therefore, viewing one's own wall with only statuses is much faster than viewing a friend's wall with only statuses. Viewing one's own wall with statuses and posts involves ABDecryption for the posts from friends and only AESDecryption for the statuses. The same applies to comments from friends. When a user has not posted anything herself to her friends' walls, then viewing friends' walls involves ABDecryption for each item on the wall, and so represents the worst case scenario.

The current view time (e.g., 90 s to view a user's own wall with 20 statuses,

20 posts, and 20 comments) may appear large; however, content can be displayed progressively, and thus older messages can be fetched while the user is reading the most recent messages, which are loaded within the first few seconds. We are also currently working on cryptographic optimizations to speed up these operations.

View Newsfeed: We test our prototype to evaluate the basic newsfeed functionality. This approach fetches the latest status from each of a user's friends. Figure 4.2c shows the results. An example newsfeed with 40 feeds takes around 215 s to construct and view. The results will be improved with parallel lookups and decryption. However, in current OSNs a user's newsfeed generally shows 20–30 posts at a time. Some techniques, such as showing feeds in blocks and pre-fetching the latest updates from friends while the user is offline, will improve the performance. We will investigate these techniques in our future evaluation.

Post and Comment: To post/comment on another user's wall, a user signs the reference to the post or comment with the append-policy signature key of the parent object, which she ABDecrypts from the parent object. The average time to post or comment is 3.94 s. The results are reasonable since a user can continue her OSN activities while the update is performed.

4.3.2 Experiments on PlanetLab

We perform the same experiments on 15 PlanetLab nodes to get an idea of DECENT's performance in a real deployment. Currently, the DHT has been implemented on PlanetLab using a Kademlia prototype extracted from the Likir implementation [47]. Figure 4.3 shows the results of our preliminary PlanetLab experiments. As expected, the time to view walls in PlanetLab

machines takes slightly longer because of network delays, such as the communication between peers and the proxy. In addition, because of node failures, a few of the users' walls could not be viewed, and in some experiments, walls were retrieved only partially. We also test the time to construct and view the newsfeed. A newsfeed with 11 feeds takes 37.3 s (95% confidence interval is $[34.4, 40.1]$ s), which closely resembles our simulation results. For improved performance and resilience, we are investigating the use of caching and replication parameters, which will be reported in subsequent work.

4.4 Leveraging Social Links to Improve Performance

The cost to view walls and newsfeeds in DECENT suffers from performance issues that arise due to the fetching and decryption of hundreds of small objects belonging to friends. We therefore propose an extension to DECENT. It adds an unstructured overlay to the base architecture in which a conventional distributed hash table is augmented with social links between users. New updates are immediately propagated to online social contacts. When an offline user comes back online a presence protocol is used to locate online contacts and query them directly for updates. Additionally, these contacts are used to retrieve cached updates from mutual contacts who are online as well as speed the discovery of other online contacts. The DHT is then used to retrieve updates that may not be cached, ensuring high availability of data.

To reduce computational overhead cached containers are stored in decrypted form and are shared with other contacts upon verifying that they satisfy the corresponding access policy; as such, containers must be decrypted only when fetched directly from the DHT. Storage nodes are trusted only to provide availability of the data with replication used to defend against node

failures or intentional misbehavior.

We develop a prototype implementation of the extended version in the FreePastry simulator. To demonstrate the functionality of existing OSNs, we also build and evaluate the newsfeed application. Our results show the importance of using social caching, which reduces the latency of displaying a newsfeed from hundreds of seconds in the base architecture to less than 10. Our architecture thus demonstrates how a careful combination of several distributed systems and cryptographic techniques can be used to provide a compelling privacy-preserving alternative to centralized OSNs. For further details, please refer to [37].

4.5 Conclusion

In this work we proposed DECENT, a design for decentralized social networks with an emphasis on security and privacy. DECENT uses an efficient cryptographic mechanism for confidentiality, combining traditional and advanced cryptographic schemes for integrity, and the use of a DHT for availability. We discussed the architecture in detail, and presented a prototype of our design. Simulation and experiments on PlanetLab with our prototype show that a privacy-enhanced OSN based on DHT with focus on confidentiality and integrity is a feasible architecture. Our extension to DECENT shows that a hybrid architecture that combines structured storage with unstructured overlay can avoid unnecessary decryption and therefore improve the overall performance.

5 ANONYMOUS AUDIT

Decentralization enhances security and privacy of user data by removing control over data from the social networking provider (for example, Facebook, Google+) to the users, but it introduces new challenges in security and privacy. Encrypted data stored in untrusted nodes prevents unauthorized access. Storage nodes cannot relate encrypted data to its owner because of randomized write authorization key as mentioned earlier in section 4. However, certain information, such as access patterns, are still available to the storage provider and it can tell, for example, who you interact with socially. At the same time, cryptographic access control makes it difficult to implement important functionality and security protections, such as allowing access to *friends of friends* and revoking access or maintaining an audit log of accesses. Therefore, encrypting the data can solve part of the problem, as the data owner can control access by distributing cryptographic keys to appropriate users.

In our work, we allow a data owner to receive a log of data accesses, while at the same time hiding this information from the storage provider [1]. The use of anonymization tools, such as Tor [77], for reaching the storage service can help with the latter. To address the auditability, we make use of anonymous credentials with revocable anonymity [38]. These credentials allow users to prove that they are authorized to access a service or a piece of data without

[1]Note that, though we have discussed formation of a distributed hash table by the social networking users for data storage, users may also have an option to store their data on cloud storage services.

revealing the contents of the credential and, importantly, their identity. Each access, however, generates an encrypted record that can be used by a special anonymity revocation service to learn who was behind such an access. In our architecture, this service is run by the data owner, who is able to decrypt the audit log records and get an access log.

We extend the revocable anonymous credential scheme to support credential chains that allow a credential owner delegate some or all of its access rights to someone else by creating a signed credential. We adapt a technique in anonymous webs of trust (AWoT) [39] that is able to verify signatures on credentials using *zero-knowledge proofs*. Credential chains can enable access by friends of friends, or people even further away in the social graph. Users issue certificates to their friends, who on the other hand can certify their friends. A user who wishes to access data can prove that there exists a chain of credentials that authorizes access for the user without revealing any information about the credential chain itself. At the same time, the user generates an encrypted record containing the information in the chain and proves, once again in zero knowledge, that it is constructed correctly. This record is then kept by the storage provider as a private access log for later decryption and verification by the data owner.

We implement the proposed scheme and develop a prototype as a proof of concept. We use Σ protocol [78] and zero knowledge proofs [79]. Experiment with our prototype shows that creating proofs and verifying them are expensive (specifically in terms of time required). However, replacing our signature and zero knowledge proof schemes with other signature and zero knowledge proof techniques may improve our results.

5.1 Background

5.1.1 Commitment

Commitment is a cryptographic scheme that allows a party to commit to a value while having the capability to hide it and reveal it later. Commitment consists of two phases – 1) commit and 2) reveal. Commitment is *hiding*, that means the chosen value to commit is hidden during the commit phase, and *binding*, meaning the committed value cannot be changed. Pedersen commitment [80] is information theoretically hiding and binding in discrete log assumption.

To commit to a value $x \in Z_p$, a party chooses a group $\mathbb{G}$ with large prime order p and random generator g. She chooses another random generator h of $\mathbb{G}$ where $\log_g^h$ is unknown and computing discrete logarithms is infeasible, randomly chooses $r \in Z_p$, and calculates the commitment $C = g^x h^r \mod p$. In the reveal phase, she sends x and r to V who recomputes the commitment and compares it to C. Here, p, is a *safe prime*, i.e., $p = 2q + 1$ where q is also a prime.

5.1.2 Zero Knowledge Proof

Zero knowledge proof (ZKP) of knowledge [79] is a cryptographic scheme that allows a prover P to prove a statement S to a verifier V without revealing S to V. Besides, S cannot be forged. ZKPs must have three properties – 1) completeness, i.e., if S is true, an honest verifier will be convinced of this fact, 2) soundness, i.e., if S is false, then no adversary can convince it to an honest verifier, and 3) zero-knowledge, i.e., if S is true, an adversary learns nothing else other than this fact. Generally, ZKP is a three step process where in the

second step V sends a challenge to P, which P responds to. Another form of ZKP known as Non-interactive Zero Knowledge Proof (NIZK) [81] allows the prover to send a single statement that the verifier can use to convince himself of S.

Generally, the statement to prove is of the form $y = g^x$, i.e., proving the knowledge of integer x to the base g, where $\mathbb{G}$ is a group of large prime order p , and g is a generator of $\mathbb{G}$. The prover P chooses random $r \in_R Z_p$, calculates $t = g^r$, and sends it to the verifier V. V picks a random challenge $c \in_R \{0,1\}^k$ and sends it to P. P then calculates $s = r - cx$ and sends it to V. V is convinced if $t = y^c g^s$ holds. In NIZK, the challenge is calculated by the prover as the hash of the commitments (y and t) and sent to V. V can verify whether the challenge is correct by recomputing the hash. Adopting the notation in [82], this protocol is denoted by $PK\{(\alpha) : \ y = g^\alpha\}$, where PK is an abbreviation of *proof of knowledge*. Greek letters represent values that are hidden.

5.1.3 Anonymous Credentials

Protecting user privacy while allowing them access to information is challenging. Anonymous credential systems or pseudonyms [83] allow users to prove their credentials without revealing anything about their identity. In anonymous credential systems, a user gets a certificate with a set of credentials or attributes from an organization, which she can later prove to the same or a different organization anonymously.

The desirable properties of anonymous credentials are: anonymous credentials should be unlinkable, i.e., users can use the same certificate without being linked between each use; collusion resistant, i.e., credentials from differ-

ent users cannot be combined together; private, i.e., nothing else other than the user owning a set of credentials is revealed; prevent against misuse, i.e., credentials cannot be used for any other reason than what they have been assigned for. Also, delegatable credentials are appropriate for most cases since it allows a certificate on a set of attributes from a party u to v to be delegated to another party w. w can then anonymously prove to u that she has a certificate that starts from u and ends at herself.

5.1.4 Anonymous Webs of Trust

Certificates bind a public key to its owner. When a party u signs the public key of another party v, he certifies that he trusts u. Web of trust allows to put trust in a public key through a chain of certificates. Each user has a public key pk and a secret key sk. A certificate chain $\texttt{sig}(pk_3, sk_2)\texttt{, sig}(pk_2, sk_1)$ means that the owner of pk_1 has certified pk_2, which certifies pk_3. If a party gets a message m signed by a public key pk and cannot find a direct certificate for pk, i.e., he did not sign that key, he looks up for a chain of certificates that starts with him and ends with pk.

Anonymous web of trust [39] allows to prove a chain of certificates anonymously using zero knowledge proof. The only information that is revealed is the length of the certificate chain. In its simplest form, the certificate contains a signature on a public key. While extended, the certificate can show sign on attributes embedded with the public key.

5.2 Architecture

5.2.1 Certificates

Certification binds a public key to its owner, and proves a user's trust in the owner of that key. Certificates can consist of a signature on a message, a public key, or a combination of public key and attributes. It can formally be defined as $C_i = sign(M, sk_i)$ where M is a message (m), public key (pk_{i+1}) , or public key and attribute $(pk_{i+1}, attr)$. A certificate chain $C_1...C_n$ defines a trust relationship of length $n-1$ starting from the key pk_1 and ending at pk_n, i.e., owner of pk_1 signs pk_2; owner of pk_2 signs pk_3 and so on. Verification of this chain consists of verifying each of the certificate $C_i(1 \leq i \leq n)$.

In a social networking context, generally, a certificate will consist of a signature on a relationship defined through a set of attributes or credentials and a public key. For example, a user A issues a certificate that proves that user B is the owner of the public key PK_B and has the relationship *friend* with user A(Figure 5.1). Similarly user B can do so for C. Note that certificates are unidirectional. So, B has to certify A to establish a relationship. This feature allows asymmetric relationship in OSN. For example, A may certify B as boss whereas B certifies A as colleague. This concept is similar as in [39].

The functionalities for certification will consist of assigning credentials – generate and issue certificate and distribute it through some channel; and prove credentials – prove that a certificate is on a set of credentials either directly or through a chain of certificates. Note that, a certificate gives access to encrypted data and allows auditing. A needs a decryption key in addition to a certificate to actually decrypt and read a piece of data.

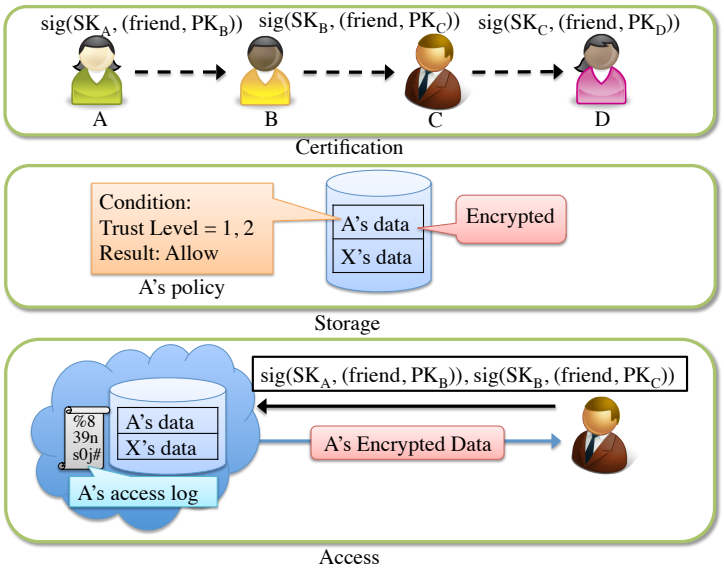


Figure 5.1: Certification, Access Control, and Private Audit in Decentralized Social Network

5.2.2 Storage Service

In a decentralized OSN, user data is distributed, and generally stored with other users in the OSN or with third party storage providers such as cloud service. In both approaches, data is encrypted to prevent unauthorized access and tampering. In addition to storing user data, the storage provider also has other responsibilities. For example,

- Availability – It ensures through replication that data is available anytime whether a user herself is online or offline.

- Access Control – Any request on data such as access, update, delete, or append operation is authenticated through certificates. The data owner attaches a policy to encrypted data that describes what trust level is required to read or write a piece of data. The storage provider uses this policy to verify the requested action on the data. However, a second level of security ensures that even if the storage provider does

not abide by the policy, the encrypted data is not visible to or updated by someone who does not have the appropriate key.

- Logging – It keeps record of each access in an encrypted access log. With anonymous audit, the storage provider does not see the record itself.

5.2.3 Anonymous Access and Auditing

Access to data is requested through a chain of certificates anonymously. The requester proves in zero knowledge that she is at a certain level of relationship d and possesses a certain set of attributes without revealing her identity. For example, she can prove that she has got the attribute *friend* from the data owner's *friend*. The storage provider verifies the proof, and after being convinced, gives the data in encrypted form to the requester.

The requester also supplies a statement that identifies her, encrypted to the owner of the access log. This information, too is proved through zero knowledge, and so, the data owner is convinced later that the access log in correct. In our current architecture, the storage provider knows what data is being accessed. However, while combined with more advanced techniques such as private information retrieval [84] , it is possible to hide this information too.

5.2.4 Combination with DECENT

While establishing relationship with a social contact by distributing secret attribute keys, users also sign the public key of the contact and grant them certificates. These certificates can be distributed through a private channel. The storage provider runs a separate service that checks the certificates in

zero knowledge and if convinced, gives the requester data that is stored encrypted with EASiER. This service also generates an access log from the zero knowledge proof that is encrypted to the data owner. The data owner can periodically get the access log and decrypt it to see who accessed her data.

In an extended version of DECENT where users cache social networking data, this approach is more useful for viewing which friends of friends or people further in relationship graph rather than direct friends accessed a piece of data as the friend information is already attached to the data itself.

5.3 Cryptographic Construction

5.3.1 Signature Scheme

We use the signature scheme by Camenisch and Lysyanskaya [85] (CL-signature) as in AWoT. CL-signature scheme consists of a public key $PK = (a, b, c, n)$ and private key $SK = p$, where $n = pq$ is a special RSA modulus (p and q are safe primes, i.e., $p = 2p' + 1$ and $q = 2q' + 1$ where p' and q' are also primes), a, b, c are randomly chosen from $\mathbb{Z}_n^*$ and a, b, c are quadratic resedue modulo n, i.e., for each of these elements (say y), $\exists x \in \mathbb{Z}_n^*$ such that $x^2 = y \mod n$.

A signature on a message m of length l_m is as follows: choose a random prime number $e > 2^{l_m+1}$ of length $l_e = l_m + 2$ and a random s of length $l_s = l_n + l_m + l$, where l is a security parameter. A value $v = (a^m b^s c)^{1/e} \mod n$ is calculated. The signature consists of the tuple (e, s, v). The verification algorithm consists of verifying $v^e = (a^m b^s c) \mod n$, and check if $2^{l_e-1} < e < 2^{l_e}$.

5.3.2 ElGamal Encryption

We use ElGamal encryption scheme [86] to create the private access log. In ElGamal scheme, a user Alice selects a group $\mathbb{G}$ of order p, a group generator g, and a random number $x \in_R \mathbb{G}$. Her public key is $(\mathbb{G}, g, p, g^x)$ and secret key is x. To encrypt a message m to Alice, Bob chooses a random $y \in_R G$, calculates g^y, computes a shared secret g^{xy}, converts m to a group member $m' \in \mathbb{G}$, and encrypts it as $c = m'g^{xy}$. Bob sends (c, g^y) to Alice. Alice computes the shared secret g^{xy} and retrieves m'. In our scheme, a message is a user's identity and credential, which she encrypts for the access log owner.

5.3.3 Protocol

To access data anonymously, a prover constructs a proof statement that consists of a signature on M. As mentioned earlier, M can be one of three forms.

Signature on a Message

The signature public key is $(\alpha = a, \beta = b, \gamma = c, \eta = n)$, the signature is (μ, ν, σ), and the intermediate values are $\tau_1 = a^m \ mod \ n, \tau_2 = b^s \ mod \ n, \tau_3 = v^e \ mod \ n, \tau_4 = a^m b^s \ mod \ n$. The verification consists of checking if $v^e = a^m b^s c \ mod \ n$, and that m and e are in proper ranges.

$$PK\{\alpha, \alpha_r, \beta, \beta_r, \gamma, \gamma_r, \eta, \eta_r, \mu, \mu_r, \nu, \nu_r, \sigma, \sigma_r, \epsilon, \epsilon_r, (\tau_i, \tau_{ir})_{i=1}^{4} :$$

$$C_a = g^{\alpha} h^{\alpha_r} \wedge C_b = g^{\beta} h^{\beta_r} \wedge C_c = g^{\gamma} h^{\gamma_r} \wedge C_n = g^{\eta} h^{\eta_r} \wedge C_m = g^{\mu} h^{\mu_r} \wedge$$

$$C_v = g^{\nu} h^{\nu_r} \wedge C_s = g^{\sigma} h^{\sigma_r} \wedge C_e = g^{\epsilon} h^{\epsilon_r} \wedge$$

$$C_{a^m} = g^{\tau_1} h^{\tau_{1r}} \wedge C_{b^s} = g^{\tau_2} h^{\tau_{2r}} \wedge C_{v^e} = g^{\tau_3} h^{\tau_{3r}} \wedge C_{a^m b^s} = g^{\tau_4} h^{\tau_{4r}} \wedge$$

$$\tau_1 = \alpha^{\mu} \wedge \tau_2 = \beta^{\sigma} \wedge \tau_3 = \nu^{\epsilon} \wedge \tau_4 = \tau_1 \cdot \tau_2 \wedge \tau_3 = \tau_4 \cdot \gamma \wedge$$

$$0 \leq \mu < 2^{l_m} \wedge 2^{l_m+1} < \epsilon < 2^{l_m+2}\}$$

Signature on a Public Key

The second type is verification of signature on a public key ($\alpha_2 = a_2, \beta_2 = b_2, \gamma_2 = c_2, \eta_2 = n_2$). The signer's public key is ($\alpha_1 = a_1, \beta_1 = b_1, \gamma_1 = c_1, \eta_1 = n_1$). (a_2, b_2, c_2) are tied together as $m = a_2 + 2^{l_{a_2}}(b_2 + 2^{l_{b_2}}(c_2 + 2^{l_{c_2}} n_2))\ mod\ n_2$ where l_{a_2}, l_{b_2}, l_{c_2} are lengths of a_2, b_2, and c_2 respectively. The sign on m is $v^e = a_1^m b_1^s\ mod\ n_1$ and the intermediate values are calculated as $\tau_1 = a_1^m\ mod\ n_1, \tau_2 = b_1^s\ mod\ n_1, \tau_3 = v^e\ mod\ n_1, \tau_4 = a_1^m b_1^s\ mod\ n_1, \theta_1 = (c_2 + 2^{l_{c_2}} n_2)\ mod\ n_2, \theta_2 = (b_2 + 2^{l_{b_2}}(c_2 + 2^{l_{c_2}} n_2))\ mod\ n_2$. The construction

is as follows:

$$
\begin{aligned}
PK\{&(\alpha_i, \beta_i, \gamma_i, \eta_i, \theta_i, \alpha_{ir}, \beta_{ir}, \gamma_{ir}, \eta_{ir}, \theta_{ir})_{i=1}^{2},\\
&\mu, \mu_r, \nu, \nu_r, \sigma, \sigma_r, \epsilon, \epsilon_r, (\tau_i, \tau_{ir})_{i=1}^{4}, (\phi_i, \phi_{ir})_{i=1}^{3} :\\
&(C_{a_i} = g^{\alpha_i} h^{\alpha_{ir}} \wedge C_{b_i} = g^{\beta_i} h^{\beta_{ir}} \wedge C_{c_i} = g^{\gamma_i} h^{\gamma_{ir}} \wedge C_{n_i} = g^{\eta_i} h^{\eta_{ir}})_{i=1}^{2} \wedge\\
&C_m = g^{\mu} h^{\mu_r} \wedge C_{c_2 n_2} = g^{\theta_1} h^{\theta_{1r}} \wedge C_{b_2 c_2 n_2} = g^{\theta_2} h^{\theta_{2r}}\\
&C_v = g^{\nu} h^{\nu_r} \wedge C_s = g^{\sigma} h^{\sigma_r} \wedge C_e = g^{\epsilon} h^{\epsilon_r} \wedge\\
&C_{a_1^m} = g^{\tau_1} h^{\tau_{1r}} \wedge C_{b_1^s} = g^{\tau_2} h^{\tau_{2r}} \wedge C_{v^e} = g^{\tau_3} h^{\tau_{3r}} \wedge C_{a_1^m b_1^s} = g^{\tau_4} h^{\tau_{4r}} \wedge\\
&(C_{n_2})^{2^{l_{c_2}}} = g^{\phi_1} h^{\phi_{1r}} \wedge (C_{c_2 n_2})^{2^{l_{b_2}}} = g^{\phi_2} h^{\phi_{2r}} \wedge (C_{b_2 c_2 n_2})^{2^{l_{a_2}}} = g^{\phi_3} h^{\phi_{3r}}\\
&\tau_1 = \alpha_1^{\mu} \wedge \tau_2 = \beta_1^{\sigma} \wedge \tau_3 = \nu^{\epsilon} \wedge \tau_4 = \tau_1 \cdot \tau_2 \wedge \tau_3 = \tau_4 \cdot \gamma_1 \wedge\\
&\theta_1 = \gamma_2 + 2^{l_{c_2}} \eta_2 \wedge \theta_2 = \beta_2 + 2^{l_{b_2}} \theta_1 \wedge \mu = \alpha_2 + 2^{l_{a_2}} \theta_2 \wedge\\
&2^{l_m+1} < \epsilon < 2^{l_m+2}\}
\end{aligned}
$$

Signature on a Public Key and Attribute

The third type is verification of signature on a public key($\alpha_2 = a_2, \beta_2 = b_2, \gamma_2 = c_2, \eta_2 = n_2$) and attribute $\xi = A$. The signer's public key is ($\alpha_1 = a_1, \beta_1 = b_1, \gamma_1 = c_1, \eta_1 = n_1$). (a_2, b_2, c_2) and A are tied together as $m = A + 2^{l_A}(a_2 + 2^{l_{a_2}}(b_2 + 2^{l_{b_2}}(c_2 + 2^{l_{c_2}} n_2)))$ where l_{a_2}, l_{b_2}, l_{c_2}, l_A are lengths of a_2, b_2, c_2, and A respectively. The sign on m is $v^e = a_1^m b_1^s \ mod \ n_1$ and the intermediate values are calculated as $\tau_1 = a_1^m \ mod \ n_1, \tau_2 = b_1^s \ mod \ n_1, \tau_3 = v^e \ mod \ n_1, \tau_4 = a_1^m b_1^s \ mod \ n_1, \theta_1 = (c_2 + 2^{l_{c_2}} n_2) \ mod \ n_2, \theta_2 = (b_2 + 2^{l_{b_2}}(c_2 + 2^{l_{c_2}} n_2)) \ mod \ n_2$, and $\theta_3 = a_2 + 2^{l_{a_2}}(b_2 + 2^{l_{b_2}}(c_2 + 2^{l_{c_2}} n_2))) \ mod \ n_2$. The construction is as

follows:

$$
\begin{aligned}
&PK\{(\alpha_i, \beta_i, \gamma_i, \eta_i, \alpha_{ir}, \beta_{ir}, \gamma_{ir}, \eta_{ir})_{i=1}^{2}, (\theta_i, \theta_{ir})_{i=1}^{3},\\
&\quad \mu, \mu_r, \nu, \nu_r, \sigma, \sigma_r, \epsilon, \epsilon_r, \xi, \xi_r, (\tau_i, \tau_{ir})_{i=1}^{4}, (\phi_i, \phi_{ir})_{i=1}^{4} :\\
&\quad (C_{a_i} = g^{\alpha_i} h^{\alpha_{ir}} \wedge C_{b_i} = g^{\beta_i} h^{\beta_{ir}} \wedge C_{c_i} = g^{\gamma_i} h^{\gamma_{ir}} \wedge C_{n_i} = g^{\eta_i} h^{\eta_{ir}})_{i=1}^{2} \wedge C_A = g^{\xi} h^{\xi_r} \wedge\\
&\quad C_m = g^{\mu} h^{\mu_r} \wedge C_{c_2 n_2} = g^{\theta_1} h^{\theta_{1r}} \wedge C_{b_2 c_2 n_2} = g^{\theta_2} h^{\theta_{2r}} \wedge C_{a_2 b_2 c_2 n_2} = g^{\theta_3} h^{\theta_{3r}}\\
&\quad C_v = g^{\nu} h^{\nu_r} \wedge C_s = g^{\sigma} h^{\sigma_r} \wedge C_e = g^{\epsilon} h^{\epsilon_r} \wedge\\
&\quad C_{a_1^m} = g^{\tau_1} h^{\tau_{1r}} \wedge C_{b_1^s} = g^{\tau_2} h^{\tau_{2r}} \wedge C_{v^e} = g^{\tau_3} h^{\tau 3r} \wedge C_{a_1^m b_1^s} = g^{\tau_4} h^{\tau_{4r}} \wedge\\
&\quad (C_{n_2})^{2^{l_{c_2}}} = g^{\phi_1} h^{\phi_{1r}} \wedge (C_{c_2 n_2})^{2^{l_{b_2}}} = g^{\phi_2} h^{\phi_{2r}} \wedge\\
&\quad (C_{b_2 c_2 n_2})^{2^{l_{a_2}}} = g^{\phi_3} h^{\phi_{3r}} \wedge (C_{a_2 b_2 c_2 n_2})^{2^{l_A}} = g^{\phi_4} h^{\phi_{4r}}\\
&\quad \tau_1 = \alpha_1^{\mu} \wedge \tau_2 = \beta_1^{\sigma} \wedge \tau_3 = \nu^{\epsilon} \wedge \tau_4 = \tau_1 \cdot \tau_2 \wedge \tau_3 = \tau_4 \cdot \gamma_1 \wedge\\
&\quad \theta_1 = \gamma_2 + 2^{l_{c_2}} \eta_2 \wedge \theta_2 = \beta_2 + 2^{l_{b_2}} \theta_1 \wedge \theta_3 = \alpha_2 + 2^{l_{a_2}} \theta_2 \wedge \mu = \xi + 2^{l_A} \theta_3 \wedge\\
&\quad 2^{l_m+1} < \epsilon < 2^{l_m+2}\}
\end{aligned}
$$

Log Entry

The construction for logging and audit is as follows: x is the access log owner's ElGamal secret key, $\chi_1 = g_1$ is the ElGamal group generator, $\chi_2 = y$ is the accessor's random secret, $\chi_3 = g_1^y$ is the accessor's random public, $\chi_4 = g_1^x$ is the access log owner's public key, $\chi_5 = g_1^{xy}$ is the shared secret, and $\chi_6 = m g_1^{xy}$ is the encryption of m. χ_3 and χ_6 are logged by the storage

provider and opened later by the access log owner.

$$
\begin{aligned}
&PK\{(\chi_i, \chi_{ir})_{i=1}^{6}, \mu, \mu_r : \\
&\quad C_m = g^{\mu} h^{\mu_r} \wedge C_{g_1} = g^{\chi_1} h^{\chi_{1r}} \wedge C_y = g^{\chi_2} h^{\chi_{2r}} \wedge \\
&\quad C_{g_1^y} = g^{\chi_3} h^{\chi_{3r}} \wedge C_{g_1^x} = g^{\chi_4} h^{\chi_{4r}} \wedge C_{g_1^{xy}} = g^{\chi_5} h^{\chi_{5r}} \wedge C_{m \cdot g_1^{xy}} = g^{\chi_6} h^{\chi_{6r}} \wedge \\
&\quad \chi_3 = \chi_1^{\chi_2} \wedge \chi_5 = \chi_4^{\chi_2} \wedge \chi_6 = \mu \cdot \chi_5\}
\end{aligned}
$$

5.4 Implementation and Evaluation

We implemented the proposed scheme in Java. We measured the time required to create and verify a proof statement of each form. We used the suggested key length in [85], which is 1024 bits for each of the four public key components a, b, c, n, and 160 bit for the message m (hence, 162 bits for e and 1344 bits for s) in the signature. However, when a public key or a public key with attribute is used as a message, the message size changes, and we set it to a group member by modular operation. Since we need to prove that some variables lie within a range, our group $\mathbb{G}$ is of order $p = 2259$. $|p|$ has to be at least $2^{(2\epsilon l+5)}$ where $\epsilon > 1$ is a security parameter. We set $\epsilon = 1.1$. We chose the length of the attribute to be 1024 bits. The group for ElGamal encryption also is of size 2259 bits.

A proof construction of signature on a message requires 3 proofs for exponentiation, and 2 proofs for multiplication. A sign on public key requires 3 proofs for exponentiation, 2 proofs for multiplication, and 3 proofs for addition. A sign on a combination of public key and attribute requires 3 proofs for exponentiation, and 2 proofs for multiplication, and 4 proofs for addition. Finally, a log entry needs 2 proof for exponentiation, and 1 proofs for multiplication. However, we can reuse some commitments as they appear in

various proofs.

Sigma protocols are expensive. We had to use large group order for higher security. Also, CL signature scheme requires use of long keys. Therefore, creation and verification of proof statements are in order of hours. We realize that the time required to create and verify the proof statements is high. However, we believe that implementing the system based on other schemes such as automorphic signature [87] and Groth-Sahai zero knowledge proof [88] will improve these results.

5.5 Related Work

A feature that allows users to keep track of who viewed their social networking profile has been enabled in the OSNs LinkedIn and Orkut. Researchers have also proposed a negotiated audit in [41]. In sh@re, a user is able to view who accessed his data if he also allows logging while accessing others data. Different typed of logging are allowed. We, for the first time, allow this feature in a decentralized social network that uses cryptography to protect user data.

Research has showed that users' access behavior can change if she knows her access is being recorded [89, 90]. For example, a user may allow her co-workers to see where she checked in, but may feel different and be encouraged to change her policy if she finds out that her boss is checking her profile frequently.

Anonymous credential schemes allow users to prove a set of credentials to a verifier in zero knowledge. Camenisch et al. propose an anonymous credential scheme that provides option for anonymity revocation [38]. However, credential delegation is an issue in their approach. Scheme by Belenkiy et

al. [91] supports delegation of credentials, but lacks anonymity revocation.

Backes et al. propose Anonymous Webs of Trust [39] that allows users to prove a certificate chain in zero knowledge. The ZKP reveals the length of the certificate chain and the recipient's identity only. However, this approach does not support anonymity revocation. Our approach combines the concept of anonymous webs of trust and anonymous credentials, and augments the level of security and privacy provided by these approaches by adding an additional level of private access log.

5.6 Conclusion

Auditable anonymity allows people at a certain distance in user's social graph to access their data anonymously while revealing their identity to the data owner through a cryptographically constructed access log. While we presented this approach in the context of social networking, it is generalized and applicable in any outsourced storage that stores sensitive data and needs this extra level of privacy. We implemented this approach. However, further investigation on the cryptographic schemes such as signature and zero knowledge proof is required for performance improvement.

6 A SECURE AND PRIVATE ONLINE SOCIAL NETWORK

We have proposed three approaches that while combined together presents a security and privacy enhanced online social network.

- First, EASiER, the encryption-based access control with efficient revocation scheme provides confidentiality to the data stored by the social networking users. It allows users to explicitly control access to their data by encrypting it while providing an option to revoke access efficiently.

- Second, DECENT, presents a comprehensive design about how to combine various cryptographic schemes including EASiER for confidentiality and integrity, distributed hash table for storage and availability, and object oriented concept for data representation with flexible, expressive, and better privacy policy. An extension to DECENT [37] allows more efficient data retrieval by leveraging social trust and caching of unencrypted data.

- Finally, auditable anonymity constructs an access log for the data owner and reveals who accesses her data while hiding this information from the storage provider. The storage provider is generalized and can be a user in the social network or a cloud service. This service is run by the storage provider. While combined with DECENT, it adds an extra level of security and privacy. In the extended version of DECENT, this approach helps users identify parties who are farther than

one hop in relationship and access their data, for example access by *friend of friend*.

7 CONCLUSION

Online social networking is the next de-facto way of communication. It is even getting out of *socialization only* boundary and being used in other domains such as anonymization communication [92], routing [93], detecting sybil nodes [94], and healthcare domain. Therefore, both users and researchers are emphasizing on information security and privacy in OSNs. We believe that security and privacy in online social networking is still an emerging topic, and researchers are still working on various approaches to address this challenge. Different techniques come with their own benefits and shortcomings.

In this thesis, we have mentioned three techniques to enhance security and privacy properties of OSNs.

- First, we presented an encryption technique EASiER that supports expressiveness, fine-grained policy, and efficient revocation. This scheme provides confidentiality to the data stored by the social networking users. It allows users to explicitly control access to their data by encrypting it while providing an option to revoke access efficiently.

- Next, we presented a design and prototype of a decentralized architecture and how to address various challenges of decentralization. DECENT presents a comprehensive design about how to combine various cryptographic schemes including EASiER for confidentiality and integrity, distributed hash table for storage and availability, and ob-

ject oriented concept for data representation with flexible, expressive, and better privacy policy. An extension to DECENT [37] allows more efficient data retrieval by leveraging social trust and caching of unencrypted data.

- Finally, we presented a cryptographic technique to allow access trace in a secure environment while hiding this information from the storage provider. Auditable anonymity constructs an access log for the data owner and reveals who accesses her data while hiding this information from the storage provider. The storage provider is generalized and can be a user in the social network or a cloud service. This service is run by the storage provider. While combined with DECENT, it adds an extra level of privacy that gives the data owner an implication of her privacy policy.

Our future direction of research includes application of a secure and private social network in healthcare domain. We envision a social networking system that users can use without putting absolute trust in a single and powerful authority. We also see a future where a security and privacy enhanced social networking will be used in providing better service to the general people.

REFERENCES

[1] "Facebook's latest news, announcements and media resources," http://newsroom.facebook.com.

[2] T. Watkins, "Suddenly, Google Plus Is Outpacing Twitter To Become The World's Second Largest Social Network," in *Business Insider*, May 2013.

[3] H. Tsukayama, "Twitter turns 7: Users send over 400 million tweets per day," in *The Washington Post*.

[4] "LinkedIn Facts," http://press.linkedin.com/about/.

[5] "Stats: How The Top Social Networks Are Growing," in *tweetsmarter*.

[6] A. Picard, "The History of Twitter, 140 Characters at a Time," in *The Globe and Mail*.

[7] "Please Rob Me," http://pleaserobme.com/.

[8] T. Pontes, G. Magno, M. Vasconcelos, A. Gupta, J. Almeida, P. Kumaraguru, and V. Almeida, "Beware of What You Share: Inferring Home Location in Social Networks," *ICDMW*, 2012.

[9] A. Acquisti and R. Gross, "Predicting Social Security numbers from public data," *National Academy of Sciences*, 2009.

[10] R. Gross and A. Acquisti, "Information Revelation and Privacy in Online Social Networks (The Facebook Case)," in *WPES*, 2005.

[11] T. N. Jagatic, N. A. Johnson, M. Jakobsson, and F. Menczer, "Social Phishing," *CACM*, vol. 50, no. 10, 2007.

[12] E. Mills, "Facebook Hit by Phishing Attacks for a Second Day," in *CNN*.

[13] M. Cha, A. Mislove, B. Adams, and K. P. Gummadi, "Characterizing Social Cascades in Flickr," in *WOSN*, 2008.

[14] "More Advertising Issues on Facebook," http://theharmonyguy.com/2008/06/20/more-advertising-issues-on-facebook/.

[15] M. O'Connor, "Facebook Revealed Private Email Addresses Last Night," in *GAWKER*, 2010.

[16] "Facebook in Privacy Breach," http://online.wsj.com/article/SB10001424052702304772804575558484075236968.html.

[17] E. Mills, "Facebook Suspends App that Permitted Peephole," in *cnet*, 2008.

[18] S. Guha, K. Tang, and P. Francis, "NOYB: Privacy in Online Social Networks," in *WOSN*, 2008.

[19] M. M. Lucas and N. Borisov, "FlyByNight: Mitigating the Privacy Risks of Social Networking," in *WPES*, 2008.

[20] K. B. Frikken, "Key Allocation Schemes for Private Social Networks ," *Social Networks*, pp. 11–19, 2009.

[21] J. Anderson, C. Diaz, J. Bonneau, and F. Stajano, "Privacy-Enabling Social Networking Over Untrusted Networks," in *WOSN*, 2009.

[22] A. Shakimov, A. Varshavsky, L. Cox, and R. Caceres, "Privacy, Cost, and Availability Tradeoffs in Decentralized OSNs," *WOSN*, 2009.

[23] L. Aiello and G. Ruffo, "LotusNet: Tunable Privacy for Distributed Online Social Network Services," *Computer Communications*, vol. 35, no. 1, pp. 75–88, 2012.

[24] S. Buchegger, D. Schiöberg, L. H. Vu, and A. Datta, "PeerSoN: P2P Social Networking — Early Experiences and Insights," in *SNS*, 2009.

[25] L. A. Cutillo, R. Molva, and T. Strufe, "Safebook: Feasibility of Transitive Cooperation for Privacy on a Decentralized Social Network," in *WOWMOM*, 2009.

[26] D. Grippi, M. Salzberg, R. Sofaer, and I. Zhitomirskiy, "DIASPORA*," https://joindiaspora.com/.

[27] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin, "Persona: An Online Social Network with User-defined Privacy," in *ACM SIGCOMM*, 2009.

[28] "Privacy Policy, PatientsLikeMe."

[29] R. Waters, "Google Seeks to Quell Buzz Privacy Outcry," Financial Times, Feb. 2010.

[30] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach, "Secure Routing for Structured Peer-to-Peer Overlay Networks," in *OSDI*, 2002.

[31] C. Lesniewski-Laas and M. F. Kaashoek, "Whanau: a Sybil-proof distributed hash table," in *NSDI*, 2010.

[32] D. Dumitriu, E. Knightly, A. Kuzmanovic, I. Stoica, and W. Zwaenepoel, "Denial-of-service Resilience in Peer-to-peer File Sharing Systems," in *ACM SIGMETRICS*, 2005.

[33] I. Osipkov, P. Wang, and N. Hopper, "Robust Accounting in Decentralized P2P Storage Systems," in *ICDCS*, 2006.

[34] S. Jahid, P. Mittal, and N. Borisov, "EASiER: Encryption-based Access Control in Social Networks with Efficient Revocation," in *ASIACCS*, 2011.

[35] S. Jahid and N. Borisov, "PIRATTE: Proxy-based Immediate Revocation of ATTribute-based Encryption," *CoRR*, 2012.

[36] S. Jahid, S. Nilizadeh, P. Mittal, N. Borisov, and A. Kapadia, "DECENT: A Decentralized Architecture for Enforcing Privacy in Online Social Networks," in *SESOC*, 2012.

[37] S. Nilizadeh, S. Jahid, P. Mittal, N. Borisov, and A. Kapadia, "Cachet: A Decentralized Architecture for Privacy Preserving Social Networking with Caching," in *CoNEXT '12*, 2012.

[38] J. Camenisch and A. Lysyanskaya, "An Efficient System for Nontransferable Anonymous Credentials with Optional Anonymity Revocation," in *EUROCRYPT*, 2001.

[39] M. Backes, S. Lorenz, M. Maffei, and K. Pecina, "Anonymous Webs of Trust," in *PETS*, 2010.

[40] K. Singh, S. Bhola, and W. Lee, "xBook: Redesigning Privacy Control in Social Networking Platforms," in *USENIX Security*, 2009.

[41] A. Gutierrez, A. Godiyal, M. Stockton, M. LeMay, C. A. Gunter, and R. H. Campbell, "Sh@re: Negotiated Audit in Social Networks," in *Proceedings of the 2009 IEEE international conference on Systems, Man and Cybernetics*, ser. SMC'09, 2009.

[42] E. D. Cristofaro, C. Soriente, G. Tsudik, and A. Williams, "Hummingbird: Privacy at the Time of Twitter," in *IEEE S & P*, 2012.

[43] W. Luo, Q. Xie, and U. Hengartner, "FaceCloak: An Architecture for User Privacy on Social Networking Sites," in *PASSAT*, Aug. 2009, pp. 26–33.

[44] Beato, Filipe and Kohlweiss, Markulf and Wouters, Karel, "Scramble! Your Social Network Data." in *PETS*, 2011.

[45] Feldman, Ariel J. and Blankstein, Aaron and Freedman, Michael J. and Felten, Edward W., "Social Networking with Frientegrity: Privacy and Integrity with an Untrusted Provider," in *USENIX Conference on Security Symposium*, 2012.

[46] M. Mani, A.-M. Nguyen, and N. Crespi, "SCOPE: A prototype for spontaneous P2P social networking," in *PerCom Workshops*, 2010.

[47] L. M. Aiello, M. Milanesio, G. Ruffo, and R. Schifanella, "An Identity-based Approach to Secure P2P Applications with Likir," *Peer-to-Peer Networking and Applications*, 2011.

[48] P. Maymounkov and D. Mazières, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric," in *IPTPS*. Springer-Verlag, 2002.

[49] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-Policy Attribute-Based Encryption," in *IEEE S & P*, 2007.

[50] E. Gilbert and K. Karahalios, "Predicting Tie Strength with Social Media," in *CHI* . ACM, 2009.

[51] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, "Measurement and Analysis of Online Social Networks," in *IMC*. ACM, 2007.

[52] M. Naor and B. Pinkas, "Efficient Trace and Revoke Schemes," in *FC*, 2001.

[53] A. Sahai and B. Waters, "Fuzzy Identity Based Encryption," in *Eurocrypt*, 2005.

[54] V. Goyal, O. Pandey, A. Sahai, and B. Water, "Attribute-based Encryption for Fine-grained Access Control of Encrypted Data," in *ACM CCS*, 2006.

[55] A. Shamir, "How to Share a Secret," *CACM*, 1979.

[56] L. Ballard, M. Green, B. de Medeiros, and F. Monrose, "Correlation-resistant storage via keyword-searchable encryption," ePrint, Tech. Rep. 2005/417, 2005.

[57] J. Bethencourt, A. Sahai, and B. Waters, "CP-ABE Toolkit," http://acsc.cs.utexas.edu/cpabe/.

[58] A. Miyaji, M. Nakabayashi, and S. TAKANO, "New Explicit Conditions of Elliptic Curve Traces for FR-Reduction," 2001.

[59] N. Attrapadung and H. Imai, "Conjunctive Broadcast and Attribute-Based Encryption," in *Pairing*, 2009.

[60] A. Boldyreva, V. Goyal, and V. Kumar, "Identity-based Encryption with Efficient Revocation," in *ACM CCS*, 2008.

[61] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-based Encryption with Non-monotonic Access Structures," in *ACM CCS*, 2007.

[62] A. Lewko, A. Sahai, and B. Waters, "Revocation Systems with Very Small Private Keys," in *IEEE S & P*, 2010.

[63] J. Hur, "Improving Security and Efficiency in Attribute-Based Data Sharing," *IEEE TKDE*, 2011.

[64] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute Based Data Sharing with Attribute Revocation," in *ASIACCS* , 2010.

[65] M. Blaze, G. Bleumer, and M. Strauss, "Divertible Protocols and Atomic Proxy Cryptography," in *EUROCRYPT*, 1998.

[66] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved Proxy Re-encryption Schemes with Applications to Secure Distributed Storage," in *NDSS*, 2005.

[67] R. Canetti and S. Hohenberger, "Chosen-ciphertext Secure Proxy Re-encryption," in *ACM CCS*, 2007.

[68] B. Libert and D. Vergnaud, "Unidirectional Chosen-ciphertext Secure Proxy Re-encryption," in *PKC*, 2008.

[69] M. Green and G. Ateniese, "Identity-Based Proxy Re-encryption," in *ACNS*, 2007.

[70] X. Liang, Z. Cao, H. Lin, and J. Shao, "Attribute Based Proxy Re-encryption with Delegating Capabilities," in *ASIACCS*, 2009.

[71] A. Rowstron and P. Druschel, "Pastry: Scalable Dstributed Object Location and Routing for Large-scale Peer-to-peer Systems," in *Middleware*, 2001.

[72] P. Maymounkov and D. Mazières, "Kademlia: A Peer-to-Peer Information System Based on the XOR Metric," in *IPTPS*, 2002.

[73] R. Baden, A. Bender, N. Spring, B. Bhattacharjee, and D. Starin, "Persona: An Online Social Network with User-defined Privacy," in *ACM SIGCOMM*, 2009.

[74] Z. Wilcox-O'Hearn and B. Warner, "Tahoe: The Least-authority Filesystem," in *StorageSS*, 2008.

[75] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "Recommendation for Key Management—Part 1: General (Revised)," NIST, Tech. Rep. SP800–57, 2007.

[76] L. M. Aiello, M. Milanesio, G. Ruffo, and R. Schifanella, "Tempering Kademlia with a robust identity based system," in *P2P*, 2008.

[77] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation Onion Router," in *USENIX Security*, 2004, pp. 303–320.

[78] R. Cramer, I. Damgard, and B. Schoenmakers, "Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols," in *CRYPTO*, 1994.

[79] Goldwasser, S. and Micali, S. and Rackoff, C., "The knowledge complexity of interactive proof systems," *SIAM J. Comput.*, 1989.

[80] T. P. Pedersen, "Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing," in *CRYPTO*, 1992.

[81] M. Blum, P. Feldman, and S. Micali, "Non-interactive zero-knowledge and its applications," in *Proceedings of the twentieth annual ACM symposium on Theory of computing*, ser. STOC '88, 1988.

[82] J. Camenisch and M. Stadler, "Efficient Group Signature Schemes for Large Groups," in *CRYPTO*, 1997.

[83] D. Chaum, "Security without identification: transaction systems to make big brother obsolete," *Commun. ACM*, 1985.

[84] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan, "Private information retrieval," in *FOCS*, 1995.

[85] J. Camenisch and A. Lysyanskaya, "A Signature Scheme with Efficient Protocols," in *SECURECOMM*, 2003.

[86] T. El Gamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Proceedings of CRYPTO 84 on Advances in cryptology*, 1985.

[87] M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev, and M. Ohkubo, "Structure-preserving signatures and commitments to group elements," in *Proceedings of the 30th annual conference on Advances in cryptology*, 2010.

[88] J. Groth and A. Sahai, "Efficient Non-interactive Proof Systems for Bilinear Groups," in *EUROCRYPT*, 2008.

[89] R. Schlegel, A. Kapadia, and A. J. Lee, "Eyeing Your Exposure: Quantifying and Controlling Information Sharing for Improved Privacy," in *SOUPS*, 2011.

[90] Y. L. Gall, A. J. Lee, and A. Kapadia, "PlexC: A Policy Language for Exposure Control," in *Proceedings of The 17th ACM Symposium on Access Control Models and Technologies (SACMAT)*, 2012.

[91] M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, and H. Shacham, "Randomizable Proofs and Delegatable Anonymous Credentials," in *CRYPTO*, 2009.

[92] P. Mittal, M. Wright, and N. Borisov, "Pisces: Anonymous Communication Using Social Networks," *NDSS*, 2013.

[93] P. Mittal, M. Caesar, and N. Borisov, "X-Vine: Secure and Pseudonymous Routing Using Social Networks," 2012.

[94] G. Danezis and P. Mittal, "SybilInfer: Detecting Sybil Nodes using Social Networks," in *NDSS*, 2009.